

PYTHON MANUAL BOOK

For Beginners



By

Tin Mai Zaw

2025 EDITION
Northern City, Yangon, Myanmar
©2025 Northern City. All Rights Reserved





အမှာစကား

ဤစာအုပ်သည် Python Programming ကို အခြေခံမှစတင် လေ့လာမည့် သူများအတွက် လက်စွဲစာအုပ်ဖြစ်ပါတယ်။ ဒုတိယအကြိမ် မြန်မာလို ထုတ်ဝေခြင်းဖြစ်ပါတယ်။ Python Reference Books, Online Reference Website တွေအပြင် Chat GPT, Deepseek AI tools တွေသုံးပြီး အချို့သင်ခန်းစာတွေကို ထပ်မံ ဖြည့်စွက်ထားပါတယ်။ ကွန်ပျူတာကျောင်း သူကျောင်းသားများ၊ အင်ဂျင်နီယာအိုင်တီ ကျောင်းသူကျောင်းသားများကို တစ်ဖက်တစ်လမ်း အထောက်အကူပြုစေရန်အတွက် ရည်ရွယ် ထုတ်ဝေရခြင်းလည်းဖြစ်ပါတယ်။ အိုင်တီနဲ့ အသက်မွေးဝမ်းကြောင်းပြုလိုသူများ၊ software programmer ဖြစ်ချင်သူများ၊ ကမ္ဘာကျော် software application တွေတီထွင်ပြီး millionaire သူဌေးဖြစ်ဖို့ စိတ်ကူးအိမ်မက်ရှိသူ မည်သူမဆို လွယ်လွယ်ကူကူ လေ့လာနိုင်အောင် လေ့ကျင့်ခန်း ဥပမာများ ၊ သင်ခန်းစာ video tutorial များနဲ့ တွဲပြီးတော့ ပြည့်ပြည့်စုံစုံ ရေးသားဖန်တီးထားတဲ့ စာအုပ်ဖြစ်ပါတယ်။ ဘွဲ့လွန် Master ၊ Ph.d တက်နေသော သူများ၊ ကိုယ်ပိုင် စာတမ်းပြုစုနေသူများ အတွက်လည်း data analysis foundation တွေကို လေ့လာနိုင်အောင် numpy ၊ pandas နဲ့ matplotlib လေ့ကျင့်ခန်းတွေကို အလွယ်ကူဆုံး နားလည်နိုင်အောင် ရေးသားဖော်ပြထားတဲ့ စာအုပ်တစ်အုပ်ဖြစ်ပါတယ်။



မာတိကာ

စာမျက်နှာ

အကြောင်းအရာ

3	Python သမိုင်းအကျဉ်း
4	ဘာလို့ python လေ့လာသင့်သလဲ
5	အခန်း (၁) – Software Installation
14	အခန်း(၂) – Python Data Types
21	အခန်း(၃) – Basic Calculations
30	အခန်း (၄) – Conditional Statements
42	အခန်း (၅) – Loops
57	အခန်း (၆) – Methods
71	အခန်း (၇) – args & kwargs
79	အခန်း(၈) – built-in data structures
116	အခန်း(၉)- List Comprehensions
123	အခန်း (၁၀) – Lambda, Map and Filter
136	အခန်း (၁၁) – Random Numbers & String
142	အခန်း (၁၂) – Regular Expressions
157	အခန်း(၁၃) – OOP
165	အခန်း(၁၄) - Interface
172	အခန်း(၁၅) – File Handling
187	အခန်း(၁၆) – Jupyter Notebook Installation
198	အခန်း(၁၇) – Numpy
217	အခန်း(၁၈) – Pandas
239	အခန်း(၁၉) – Matplotlib
253	အခန်း(၂၀) – Seaborn
266	အခန်း(၂၁) – Minigames
290	အခန်း(၂၂) – Python & MySQL Database



Python သမိုင်းအကျဉ်း

Python programming language ၏ သမိုင်းကြောင်းကို 1990 ခုနှစ်များအစောပိုင်းမှ စတင်ခဲ့ပါတယ်။ Guido van Rossum ဆိုသူ Dutch programmer က 1989 ခုနှစ်၊ ခရစ္စမတ်ကာလအတွင်း Amsterdam မှာ အချိန်ဖြုန်းဖို့အတွက် ဒီ language ကို စတင်ရေးသားခဲ့ပါတယ်။ သူဟာ ABC language ကို အခြေခံပြီး ပိုမိုကောင်းမွန်တဲ့ scripting language တစ်ခုဖန်တီးဖို့ ရည်ရွယ်ခဲ့ပါတယ်။

Python ဆိုတဲ့အမည်ကို British comedy group Monty Python ရဲ့ "Monty Python's Flying Circus" ကနေ ယူထားတာဖြစ်ပါတယ်။ Guido က ဒီ comedy show ကို ကြိုက်နှစ်သက်တာကြောင့် သူ့ language ကို Python လို့ အမည်ပေးခဲ့ပါတယ်။ 1991 ခုနှစ်မှာ Python 0.9.0 version ကို ပထမဆုံးအကြိမ် release လုပ်ခဲ့ပါတယ်။ ဒီ version မှာ classes, inheritance, exception handling နဲ့ list, dict, str စတဲ့ data types တွေ ပါဝင်ခဲ့ပါတယ်။

2000 ခုနှစ်မှာ Python 2.0 version ထွက်ရှိခဲ့ပြီး list comprehensions, garbage collection system နဲ့ Unicode support တို့ ပါဝင်လာခဲ့ပါတယ်။ 2008 ခုနှစ်မှာ Python 3.0 (Python 3000 လို့လည်းခေါ်) ကို release လုပ်ခဲ့ပါတယ်။ ဒီ version မှာ backward compatibility မရှိဘဲ major changes တွေ ပြုလုပ်ထားတာကြောင့် ဈေးကွက်ထဲမှာ ကျယ်ကျယ်ပြန့်ပြန့် အသုံးပြုဖို့ အချိန်ယူရပါတယ်။

ယနေ့အချိန်မှာတော့ Python ဟာ world's most popular programming languages တစ်ခုဖြစ်လာပါတယ်။ 2020 ခုနှစ်မှာ Python 2.7 ကို officially ရပ်စဲပြီးနောက် Python 3.x versions တွေသာ ဆက်လက်ထွက်ရှိလျက်ရှိပါတယ်။ Python ရဲ့ simplicity, readability နဲ့ vast ecosystem တို့ကြောင့် web development, data science, artificial intelligence, machine learning စတဲ့နယ်ပယ်မျိုးစုံမှာ အသုံးပြုနေကြပါတယ်။

Python Software Foundation (PSF) က Python language ရဲ့ development နဲ့ promotion တို့ကို ဦးဆောင်လျက်ရှိပါတယ်။ 2021 ခုနှစ်မှာ Guido van Rossum ဟာ Microsoft တွင် engineer အဖြစ်ပါဝင်ခဲ့ပြီး Python development ကို ဆက်လက်ပံ့ပိုးနေဆဲဖြစ်ပါတယ်။ Python community ဟာ နိုင်ငံတကာမှာ ကြီးမားစွာတိုးတက်လာပြီး PyCon conference များကို နှစ်စဉ်ကျင်းပလျက်ရှိပါတယ်။



ဘာလို့ Python ကို လေ့လာသင့်သလဲ?

Python ကို လေ့လာသင့်တဲ့ အဓိက အကြောင်းရင်းမှာ ရိုးရှင်းလွယ်ကူတဲ့ syntax နဲ့ beginner-friendly ဖြစ်တာကြောင့်ပါ။ English language နဲ့ ဆင်တူတဲ့ code structure တွေကို အသုံးပြုထားတာမို့ programming အသစ်စသူတွေအတွက် အထူးသင့်တော်ပါတယ်။ ဥပမာ `print("Hello World")` ဆိုတဲ့ code ဟာ ဘာလုပ်တယ်ဆိုတာ ကြည့်ပြီးတာနဲ့ နားလည်နိုင်ပါတယ်။

Python ဟာ versatile ဖြစ်ပြီး application အမျိုးမျိုးအတွက် အသုံးပြုနိုင်ပါတယ်။ Web development (Django, Flask), data science (Pandas, NumPy), artificial intelligence (TensorFlow, PyTorch), game development (Pygame) စတဲ့ နယ်ပယ်စုံမှာ ကောင်းစွာအသုံးပြုနိုင်ပါတယ်။ ဒါကြောင့် Python တစ်ခုတည်းကို လေ့လာရုံနဲ့ အလုပ်အကိုင်အခွင့်အလမ်းများစွာ ရရှိနိုင်ပါတယ်။

Python community ကြီးမားပြီး support ကောင်းတာပါ။ မည်သည့် problem နဲ့ကြုံကြုံ Stack Overflow လို platform တွေမှာ အဖြေရှာဖို့လွယ်ကူပါတယ်။ မြန်မာနိုင်ငံမှာလည်း Python developer community တွေ တဖြည်းဖြည်းကြီးထွားလာနေပါတယ်။ PyCon Myanmar လို conference တွေလည်း ကျင်းပလာနေပါပြီ။

နောက်ထပ်အရေးကြီးတဲ့အချက်က job market တွင် demand မြင့်မားနေတာပါ။ LinkedIn ၏ 2023 report အရ Python skill ရှိသူများဟာ အလုပ်အကိုင်အခွင့်အလမ်း 25% ပိုများပါတယ်။ မြန်မာနိုင်ငံရဲ့ tech industry တွင်လည်း Python developer များကို လစာကောင်းပေးပြီး ခေါ်ယူနေကြပါတယ်။

အချုပ်အားဖြင့်ဆိုရင် Python ကို လေ့လာခြင်းဖြင့် အောက်ပါအကျိုးကျေးဇူးတွေ ရရှိနိုင်ပါတယ်။ လွယ်ကူတဲ့ learning curve၊ နယ်ပယ်စုံသုံးနိုင်မှု၊ ကောင်းမွန်တဲ့ community support နဲ့ အလုပ်အကိုင်အခွင့်အလမ်းများစွာ ရရှိနိုင်ခြင်းတို့ပဲဖြစ်ပါတယ်။ ဒါကြောင့် programming စတင်လေ့လာမည့်သူများအတွက်ရော၊ professional developer များအတွက်ပါ Python ဟာ အထူးသင့်တော်တဲ့ language တစ်ခုဖြစ်ပါတယ်။

အခန်း (၁)

Software Installations





၁။ Software Installation ပြုလုပ်ပုံအဆင့်ဆင့် –

၁.၁။ Python Installation

၁.၂။ PyCharm Installation

၁.၃။ Running the first Python App

၁.၁။ Installation Guide

- အောက်က download link ကို ခေါက်ပြီး Python SDK ကို အရင်သွင်းရပါမယ်။

<https://www.python.org/downloads/>



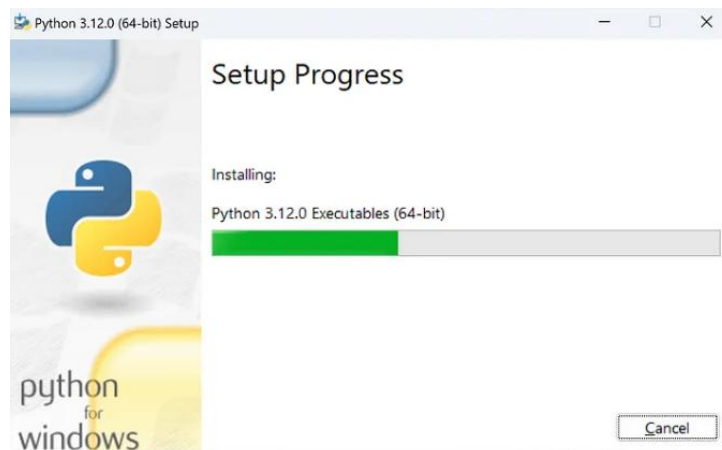
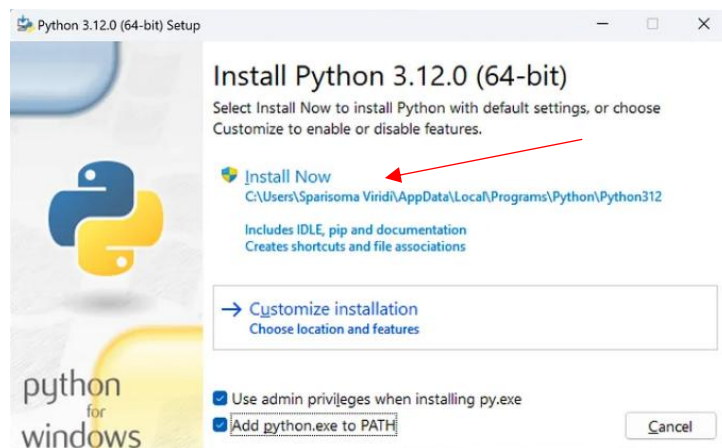
Python Installation ပြုလုပ်ပုံအဆင့်ဆင့် -

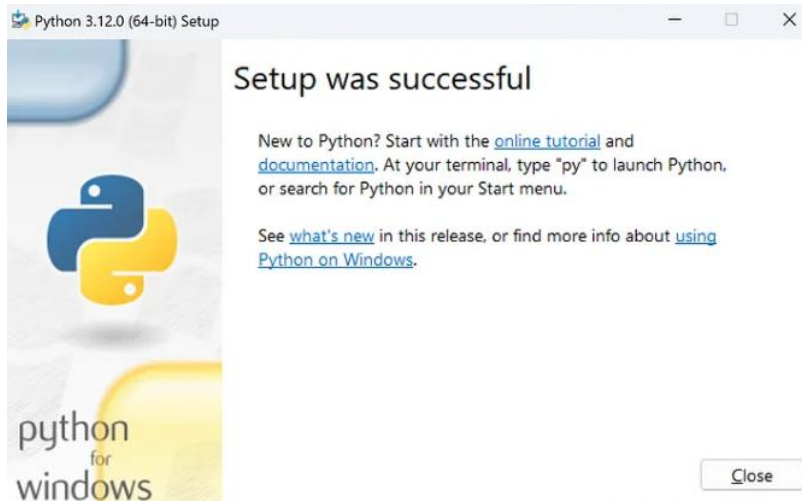


wai moe > Downloads

Search Downloads

Name	Date modified	Type	Size
▼ Today (1)			
python-3.12.3-amd64	5/31/2024 10:45 AM	Application	26,087 KB
▼ Earlier this week (6)			
360TS_Setup	5/29/2024 9:43 AM	Application	101,342 KB
360TS_Setup_Mini	5/29/2024 9:39 AM	Application	1,453 KB
7z2406-x64	5/29/2024 8:01 AM	Application	1,582 KB
os2go_4.0_portable (1)	5/28/2024 10:18 AM	Application	38,554 KB
MediaCreationTool_22H2 (1)	5/28/2024 10:17 AM	Application	19,008 KB
TS Recommended Apps	5/29/2024 9:54 AM	File folder	
▼ Last week (10)			

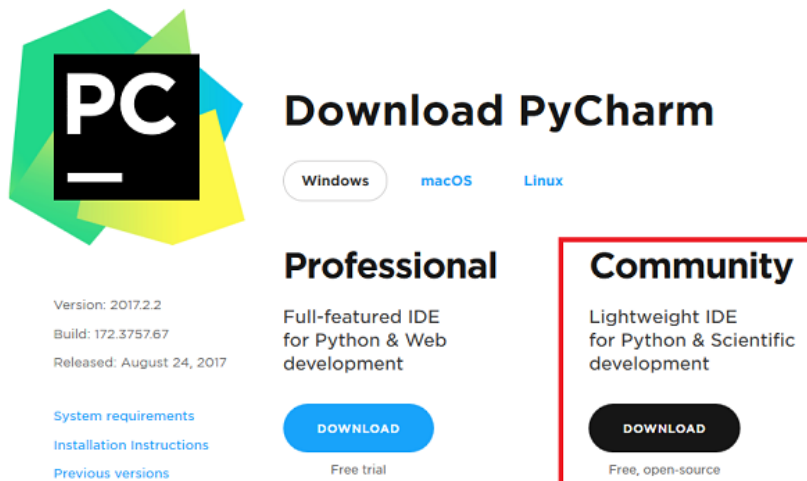




Python SDK install လုပ်တာ အောင်မြင်သွားပါပြီ။

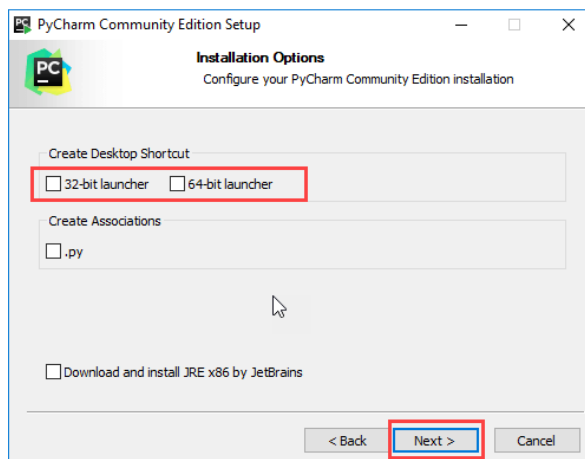
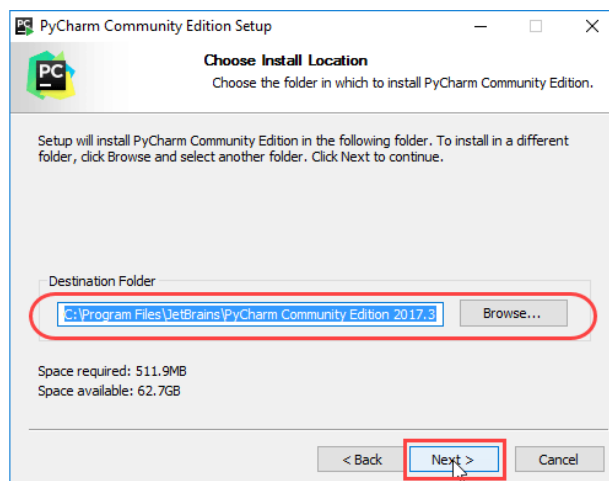
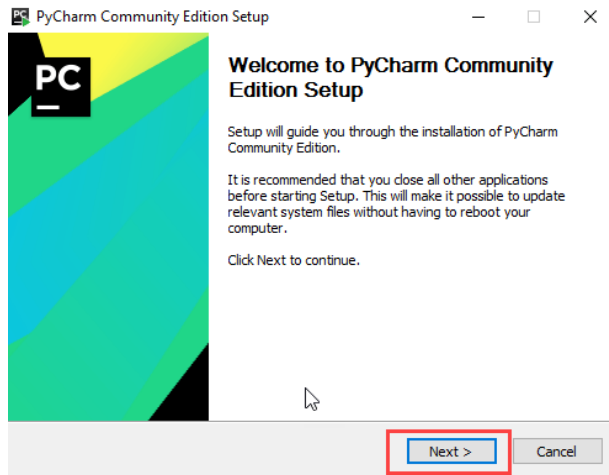
- ဒါဆို အောက်က pycharm download link ကို သွားပြီး pycharm community version ကို ဒေါင်းပါ။

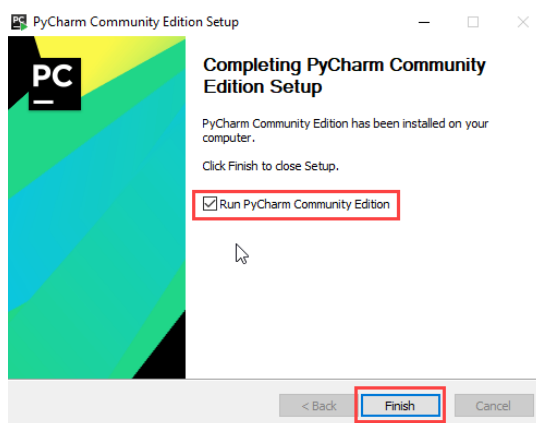
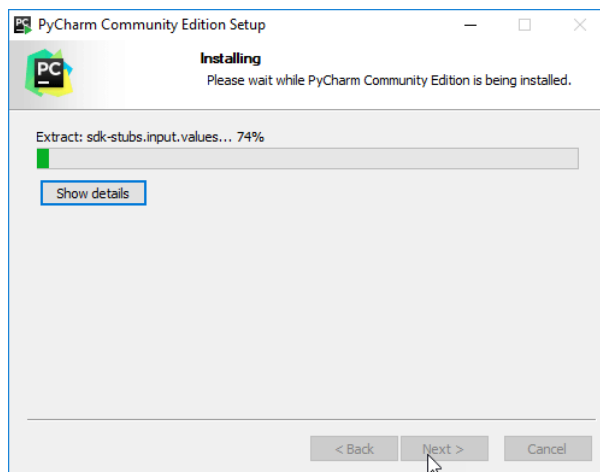
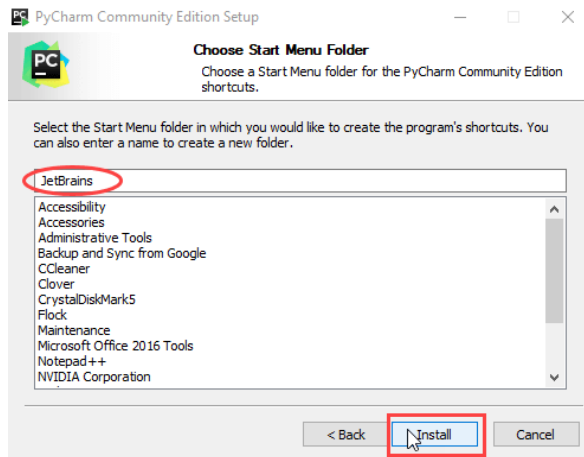
<https://www.jetbrains.com/pycharm/download/>

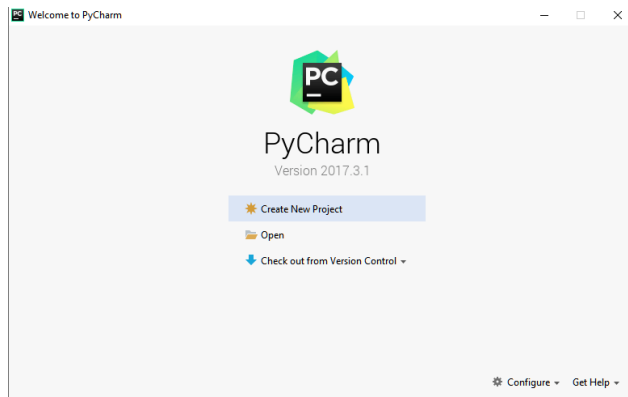


The image shows the PyCharm download page. On the left, there is a logo with 'PC' and a minus sign. Below it, the version is 2017.2.2, build is 172.3757.67, and it was released on August 24, 2017. There are links for 'System requirements', 'Installation Instructions', and 'Previous versions'. In the center, there is a 'Download PyCharm' section with buttons for 'Windows', 'macOS', and 'Linux'. Below these are two main options: 'Professional' (Full-featured IDE for Python & Web development) and 'Community' (Lightweight IDE for Python & Scientific development). The 'Community' option is highlighted with a red border. Both options have a 'DOWNLOAD' button. Below the 'Professional' button is a 'Free trial' link. Below the 'Community' button is a 'Free, open-source' link.

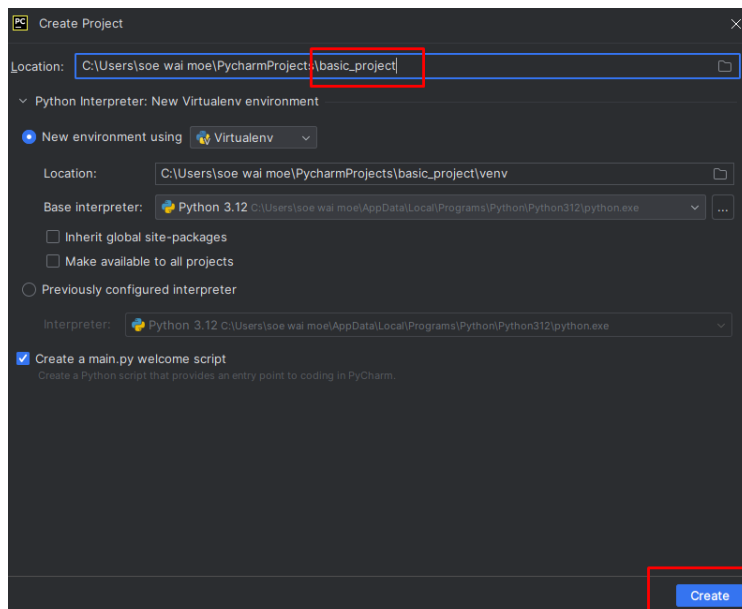
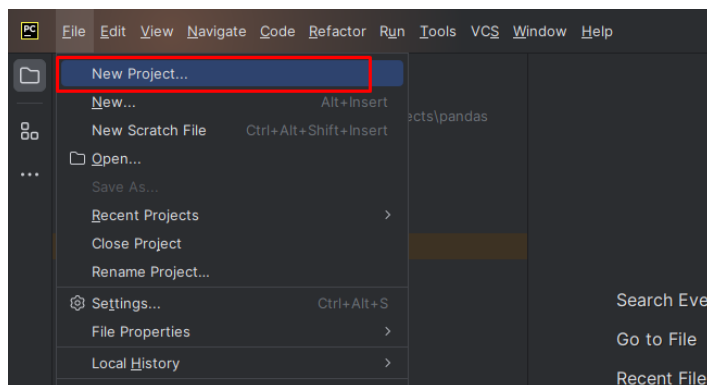
After downloading pycharm community application, you have to install pycharm app as the followings:

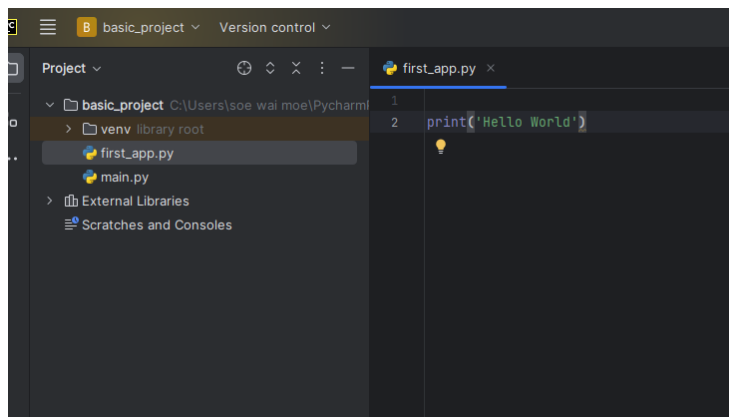
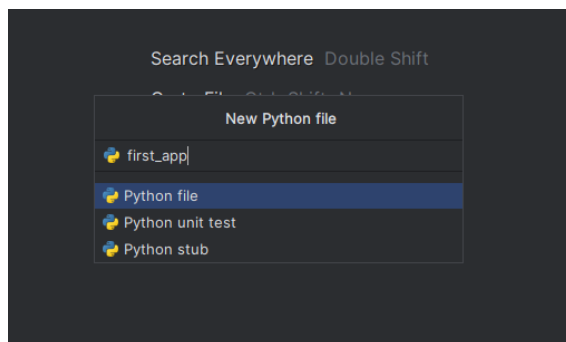
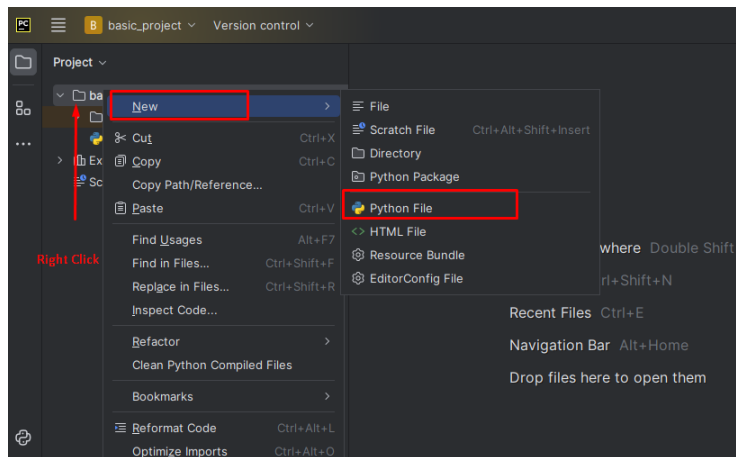


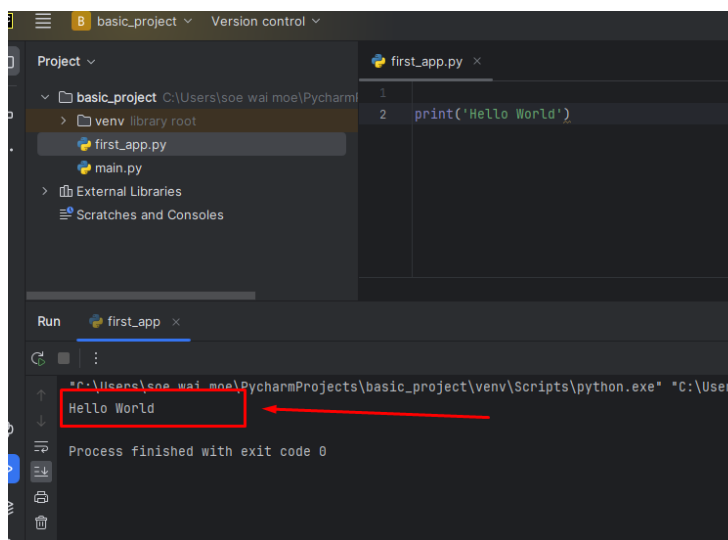
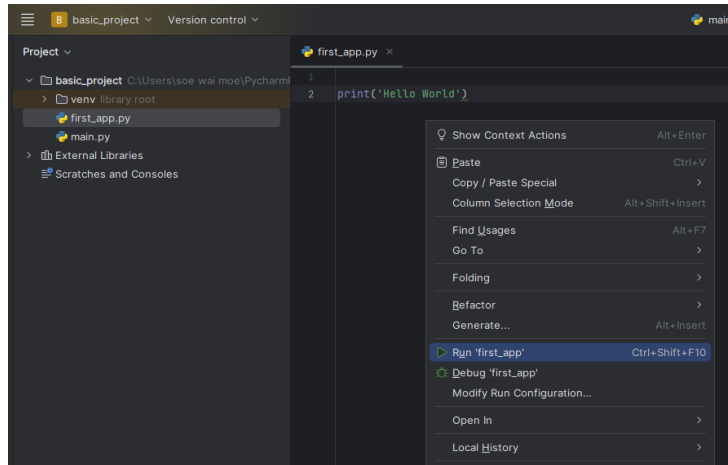




ဒီလိုပုံပေါ်လာပြီဆို pycharm installation ပြီးသွားပါပြီ။ First python program ကို run ကြည့်ဖို့ အဆင်သင့်ဖြစ်ပါပြီ။





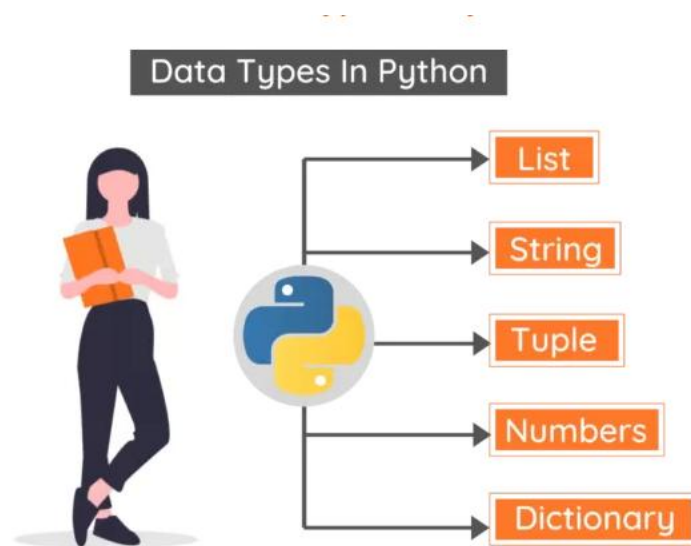


⇒ Congratulations! အခုကစပြီး Python လေ့ကျင့်ခန်းတွေကို စလုပ်လို့ရပါပြီ။



အခန်း(၂)

Python Data Types



၂။ Data Types

Data Types ဆိုတာက variable တစ်ခုမှာ သိမ်းထားတဲ့ data အမျိုးအစားကို ဖော်ပြပါတယ်။ Python မှာ Built-in Data Types အမျိုးအစား (၈) မျိုးရှိပါတယ်။ အောက်မှာ အဆင့်ဆင့်ရှင်းပြထားပါတယ်။

၂.၁။ Numeric Types (ဂဏန်းအမျိုးအစားများ)

၂.၁.၁။ Integer (int)

- ကိန်းပြည့်များအတွက်အသုံးပြုပါတယ်။

Code:

```
x = 10
y = -5
print(type(x)) # Output: <class 'int'>
```

၂.၁.၂။ Float (float)

- ဒသမကိန်းများအတွက်အသုံးပြုပါတယ်။

Code:

```
a = 3.14
b = -0.5
print(type(a)) # Output: <class 'float'>
```

၂.၁.၃။ Complex (complex)



- ကိန်းရှင်များအတွက်အသုံးပြုပါတယ်။ j ကို imaginary part အတွက်သုံးပါတယ်။

Code:

```
c = 2 + 3j
print(c.real) # Output: 2.0
print(c.imag) # Output: 3.0
```

၂.၂။ Text Type (စာသားအမျိုးအစား)

String (str)

- စာသား၊ စာကြောင်းများအတွက်အသုံးပြုပါတယ်။ " သို့မဟုတ် "'" ဖြင့်သတ်မှတ်ပါ။

Code:

```
name = "Mg Mg"
message = 'Hello, Python!'
print(name[0]) # Output: 'M' (Indexing)
print(len(message)) # Output: 13
```

၂.၃။ Boolean Type (အမှန်အမှားအမျိုးအစား)

Boolean (bool)

- True သို့မဟုတ် False တန်ဖိုးများအတွက်သုံးပါတယ်။

Code:

```
is_active = True
is_admin = False
print(5 > 3) # Output: True
```

၂.၄။ Sequence Types (အစဉ်လိုက်အမျိုးအစားများ)

၂.၄.၁။ List (list)

- Mutable (ပြင်လို့ရသော) အစုအဝေး။ [] ဖြင့်သတ်မှတ်ပါ။

Code:

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange") # ထပ်ထည့်ခြင်း
print(fruits[1]) # Output: 'banana'
```

၂.၄.၂။ Tuple (tuple)

- Immutable (ပြင်လို့မရသော) အစုအဝေး။ () ဖြင့်သတ်မှတ်ပါ။

Code:

```
colors = ("red", "green", "blue")
# colors[0] = "pink" # Error! Tuple ကို update မလုပ်နိုင်
```

၂.၄.၃။ Range (range)

- ဂဏန်းစဉ်တစ်ခုကိုကိုယ်စားပြုသည်။ Loop တွင်အသုံးများပါတယ်။

Code:

```
numbers = range(5) # 0,1,2,3,4
for num in numbers:
    print(num) # Output: 0 1 2 3 4
```

၂.၅။ Mapping Type (စာရင်းဇယားအမျိုးအစား)

Dictionary (dict)

- Key-Value စုံတွဲများကိုသိမ်းဆည်းပါတယ်။ {} ဖြင့်သတ်မှတ်ပါ။

Code:

```
user = {"name": "Alice", "age": 30}
```

```
print(user["name"]) # Output: 'Alice'  
user["email"] = "alice@example.com" # အသစ်ထည့်ခြင်း
```

၂.၆။ Set Types (အစုအဖွဲ့အမျိုးအစားများ)

၂.၆.၁။ Set (set)

- ထပ်နေသော element များမပါဝင်ပါ။ {} သို့မဟုတ် set() ဖြင့်သတ်မှတ်ပါ။

Code:

```
unique_numbers = {1, 2, 2, 3} # {1, 2, 3} အဖြစ်သိမ်းမည်  
unique_numbers.add(4)
```

၂.၆.၂။ Frozen Set (frozenset)

- Immutable version of set (ပြင်လို့မရသော set)။

Code:

```
f_set = frozenset([1, 2, 3])  
# f_set.add(4) # Error!
```

၂.၇။ None Type (ဗလာတန်ဖိုး)

NoneType (None)

- တန်ဖိုးမရှိသော variable များအတွက်သုံးပါတယ်။

Code:

```
result = None  
print(result) # Output: None
```

၂.၈။ Data Types စစ်ဆေးခြင်းနှင့် ပြောင်းလဲခြင်း

၂.၈.၁။ type() ဖြင့် Data Type စစ်ဆေးခြင်း

Code:

```
x = 5
print(type(x)) # Output: <class 'int'>
```

၂.၈.၂။ isinstance() ဖြင့် Data Type စစ်ဆေးခြင်း

Code:

```
print(isinstance(3.14, float)) # Output: True
```

၂.၈.၃။ Type Conversion (အမျိုးအစားပြောင်းလဲခြင်း)

Code:

```
num_str = "123"
num_int = int(num_str) # String ကို int ပြောင်းခြင်း
print(num_int + 5) # Output: 128
```

၂.၉။ Mutable vs Immutable Data Types

Mutable (ပြင်လို့ရ) Immutable (ပြင်လို့မရ)

List, Dict, Set Int, Float, String, Tuple

ဥပမာ:

Code:

```
# Immutable (String)
s = "hello"
# s[0] = "H" # Error! Strings are immutable

# Mutable (List)
lst = [1, 2, 3]
lst[0] = 10 # ပြင်လို့ရသည်
```



📌 Short Notes:

- အခြေခံ Data Types: int, float, str, bool
- အစုအဝေး Data Types: list, tuple, set, dict
- Type Conversion: int(), str(), list() တို့ကိုသုံးပြီး အမျိုးအစားပြောင်းနိုင်သည်။
- Mutable vs Immutable: List, Dict, Set သည် mutable ဖြစ်ပြီး၊ String, Tuple သည် immutable ဖြစ်သည်။

Python Data Types ကို ဒီနည်းနဲ့ လက်တွေ့အသုံးပြုနိုင်ပါတယ်။ လိုအပ်တဲ့ Data Type ကို ရွေးချယ်အသုံးပြုခြင်းဖြင့် Program ရဲ့ efficiency ကို မြှင့်တင်နိုင်ပါတယ်။

အခန်း(၃)

Basic Calculations



❑ Basic Calculations

Basic calculation တွေဖြစ်တဲ့ add (ပေါင်း), subtract (နှုတ်), multiply (မြှောက်), divide (စား) တွေကို လုပ်ဆောင်နိုင်ပါတယ်။ ဒီလို calculation တွေကို လုပ်ဆောင်ဖို့ Python မှာ arithmetic operators တွေကို အသုံးပြုပါတယ်။

၃.၁။ Addition (ပေါင်းခြင်း)

ပေါင်းခြင်းအတွက် + operator ကို အသုံးပြုပါတယ်။

```
python
a = 10
b = 5
result = a + b
print(result) # Output: 15
```

၃.၂။ Subtraction (နှုတ်ခြင်း)

နှုတ်ခြင်းအတွက် - operator ကို အသုံးပြုပါတယ်။

```
python
a = 10
b = 5
result = a - b
print(result) # Output: 5
```

၃.၃။ Multiplication (မြှောက်ခြင်း)

မြှောက်ခြင်းအတွက် * operator ကို အသုံးပြုပါတယ်။

```
python
a = 10
b = 5
```



```
result = a * b
print(result) # Output: 50
```

၃.၄။ Division (စားခြင်း)

စားခြင်းအတွက် / operator ကို အသုံးပြုပါတယ်။

```
python
a = 10
b = 5
result = a / b
print(result) # Output: 2.0
```

၃.၅။ Floor Division (ပြည့်ဝင်စားခြင်း)

ပြည့်ဝင်စားခြင်းအတွက် // operator ကို အသုံးပြုပါတယ်။ ဒီ operator က division လုပ်ပြီး ရလာတဲ့ result ကို integer အဖြစ် ပြန်ပေးပါတယ်။

```
python
a = 10
b = 3
result = a // b
print(result) # Output: 3
```

၃.၆။ Modulus (အကြွင်း)

အကြွင်းကို ရှာဖို့ % operator ကို အသုံးပြုပါတယ်။

```
python
a = 10
b = 3
result = a % b
print(result) # Output: 1
```


၃.၇။ Exponentiation (ထပ်ကိန်း)

ထပ်ကိန်းတွက်ဖို့ `**` operator ကို အသုံးပြုပါတယ်။

```
python
a = 2
b = 3
result = a ** b
print(result) # Output: 8
```

Example:

```
python
# Basic calculations in Python
a = 10
b = 5

# Addition
add_result = a + b
print("Addition:", add_result) # Output: 15

# Subtraction
sub_result = a - b
print("Subtraction:", sub_result) # Output: 5

# Multiplication
mul_result = a * b
print("Multiplication:", mul_result) # Output: 50

# Division
div_result = a / b
print("Division:", div_result) # Output: 2.0

# Floor Division
```

```
floor_div_result = a // b
print("Floor Division:", floor_div_result) # Output: 2

# Modulus
mod_result = a % b
print("Modulus:", mod_result) # Output: 0

# Exponentiation
exp_result = a ** b
print("Exponentiation:", exp_result) # Output: 100000
```



လေ့ကျင့်ရန် -

Practice 1:

```
# data types မပြောင်းခင်

print('Basic Calculations')
print('-----')
print('Add Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= num1 + num2

print('The answer is ',ans)
```



Output:

```
Basic Calculations
```

```
-----
```

```
Add Numbers
```

```
-----
```

```
Enter Num1:12
```

```
Enter Num2: 3
```

```
The answer is :123
```

#data type ပြောင်းပြန်

```
print('Basic Calculations')
print('-----')
print('Add Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= int(num1) + int(num2)

print('The answer is ',ans)
```



Output

Basic Calculations

Add Numbers

Enter Num1:12

Enter Num2: 3

The answer is :15

Practice 2:

```
print('Basic Calculations')
print('-----')
print('Divide Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= int(num1) / int(num2)

print('The answer is ',int(ans))
```



⇒ Ouput:

```
Basic Calculations
-----
Add Numbers
-----
Enter Num1:5
Enter Num2: 3
The answer is :1
```

Practice 3:

```
print('Basic Calculations')
print('-----')
print('Divide Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= float(num1) / float(num2)

print('The answer is ',float(ans))
```



Output:

```
Basic Calculations
```

```
-----
```

```
Add Numbers
```

```
-----
```

```
Enter Num1:5
```

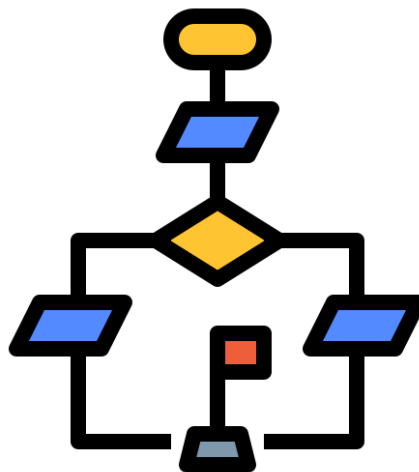
```
Enter Num2: 3
```

```
The answer is :1.6666666666666667
```



အခန်း(၄)

Conditional Statements





❑ Conditional Statements

Conditional Statements ဆိုတာက program ရဲ့ flow ကို စီမံခန့်ခွဲဖို့အတွက် အသုံးပြုပါတယ်။ ဒီ guide မှာ if, elif, else တို့ကို အသေးစိတ်ရှင်းပြပြီး example code တွေပြသပေးမှာဖြစ်ပါတယ်။

၄.၁။ if Statement

if statement က condition တစ်ခုကို စစ်ဆေးပြီး၊ condition မှန်ရင် code block ကို run ပါတယ်။

Syntax:

```
if condition:
    # Code to execute if condition is True
```

Example 1: if Statement

Code:

```
age = 18
if age >= 18:
    print("You are eligible to vote.")
```




Output:

```
You are eligible to vote.
```

၄.၂။ if-else Statement

if condition မှားရင် else block ကို run ပါတယ်။

Syntax:

```
if condition:
    # Code to execute if condition is True
else:
    # Code to execute if condition is False
```

Example 2: if-else Statement

Code:

```
age = 16
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

**Output:**

You are not eligible to vote.

၄.၃။ if-elif-else Statement

Condition အများကြီးကို စစ်ဆေးဖို့ elif (else if) ကိုသုံးပါတယ်။

Syntax:

```
if condition1:
    # Code to execute if condition1 is True
elif condition2:
    # Code to execute if condition2 is True
else:
    # Code to execute if all conditions are False
```

Example 3: if-elif-else Statement**Code:**

```
score = 85
if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Grade: D")
```



Output:

Grade: B

၄.၄။ Nested if Statements

if statement ထဲမှာ နောက်ထပ် if statement တွေထပ်သုံးနိုင်ပါတယ်။

Example 4: Nested if

Code:

```
age = 20
if age >= 18:
    if age == 18:
        print( "You just became eligible to vote." )
    else:
        print( "You are eligible to vote." )
else:
    print( "You are not eligible to vote." )
```

Output:

You are eligible to vote.



၄.၅။ Logical Operators (and, or, not)

Condition တွေကို ပေါင်းစပ်ဖို့ logical operators တွေကိုသုံးပါတယ်။

Example 5: Logical Operators

Code:

```
age = 25
```

```
is_student = True
```

```
if age >= 18 and not is_student:
```

```
    print( "You are eligible for a discount." )
```

```
else:
```

```
    print( "You are not eligible for a discount." )
```

Output:

Result:

You are not eligible for a discount.

၄.၆။ Ternary Operator

Condition ကို တစ်ကြောင်းတည်းနဲ့ ရေးချင်ရင် Ternary Operator ကိုသုံးပါတယ်။

Syntax:

```
value_if_true if condition else value_if_false
```



Example 6: Ternary Operator

Code:

```
age = 20
status = "Eligible" if age >= 18 else "Not Eligible"
print(status)
```

Output:

Eligible

၄.၇။ match-case Statement (Python 3.10+)

Python 3.10 မှာ match-case statement ကို မိတ်ဆက်ခဲ့ပါတယ်။ switch-case နဲ့ဆင်တူပါတယ်။

Syntax:

```
match value:
    case pattern1:
        # Code to execute if value matches pattern1
    case pattern2:
        # Code to execute if value matches pattern2
    case _:
        # Default case
```



Example 7: match-case Statement

Code:

```
day = "Monday"
match day:
    case "Monday":
        print("Start of the week")
    case "Friday":
        print("End of the week")
    case _:
        print("Midweek")
```

Output:

Start of the week

၄.၈။ Best Practices

1. Indentation: Python မှာ indentation ကိုသတိထားပါ။ if block ထဲက code တွေကို နေရာချပါ။
2. Readable Conditions: Condition တွေကို ရှင်းရှင်းလင်းလင်းရေးပါ။
3. Avoid Deep Nesting: Nested if တွေကို လိုအပ်မှသာသုံးပါ။

✍️ Short Notes:

1. if: Condition တစ်ခုကို စစ်ဆေးပြီး code block ကို run ပါတယ်။
2. elif: Multiple conditions တွေကို စစ်ဆေးပါတယ်။
3. else: ဘယ် condition မှမှန်ရင် run ပါတယ်။
4. Logical Operators: and, or, not တို့ကို condition တွေပေါင်းစပ်ဖို့သုံးပါတယ်။
5. Ternary Operator: တစ်ကြောင်းတည်းနဲ့ condition ရေးပါတယ်။
6. match-case: Python 3.10+ မှာ pattern matching အတွက်သုံးပါတယ်။

Python Conditional Statements ကို ဒီနည်းနဲ့ လက်တွေ့အသုံးပြုနိုင်ပါတယ်။ Program logic တွေကို ပိုမိုတိကျစွာရေးသားနိုင်ပါတယ်။



လေ့ကျင့်ရန် -

Practice 1:

```
print(' Conditional Statement ')\nprint(' Finding the largest numbers ')\nprint(' Algorithm-1 ')\nnum1=int(input('Enter Num1:'))\nnum2=int(input('Enter Num2:'))\nnum3=int(input('Enter Num3:'))
```



```
if num1>num2:  
    L=num1  
else:  
    L=num2  
  
if num3>L:  
    L=num3  
  
print('The largest number is = ', L)
```

Output:

```
Conditional Statement  
Finding the largest number  
Algorithm-1  
Enter Num1: 1  
Enter Num2: 2  
Enter Num3: 33  
The largest number is = 33
```

Practice 2:

```
print(' Conditional Statement ')  
print(' Finding the largest numbers ')  
print(' Algorithm-2 ')  
num1=int(input('Enter Num1:'))
```




```
num2=int(input('Enter Num2:'))
num3=int(input('Enter Num3:'))

if num1>num2 and num1>num3:
    L=num1
elif num2>num1 and num2>num3:
    L=num2
else:
    L=num3

print('The largest number is = ', L)
```

Output:

```
Conditional Statement
Finding the largest number
Algorithm-2
Enter Num1: 1
Enter Num2: 333
Enter Num3: 23
The largest number is = 333
```

Practice 3:

```
print(' Conditional Statement ')
print(' Finding the largest numbers ')
print(' Algorithm-3  ')
```



```
num1=int(input('Enter Num1:'))
num2=int(input('Enter Num2:'))
num3=int(input('Enter Num3:'))

if num1>num2:
    if num1>num3:
        L=num1
    else:
        L=num3
else:
    if num2>num3:
        L=num2
    else:
        L=num3

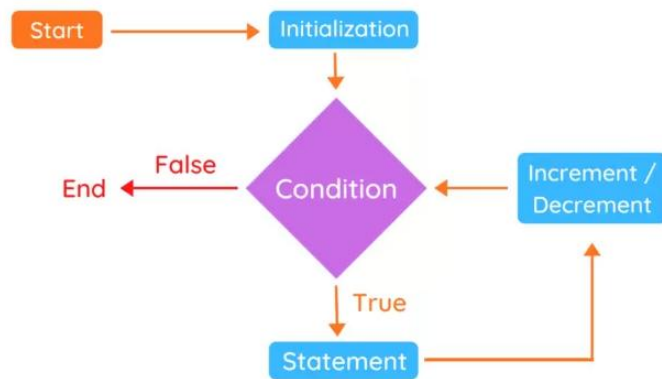
print('The largest number is = ', L)
```

Output:

```
Conditional Statement
Finding the largest number
Algorithm-3
Enter Num1: 333
Enter Num2: 22
Enter Num3: 55
The largest number is = 333
```

အခန်း(၅)

Loops





❑ Loops

Loops ဆိုတာက code block တစ်ခုကို ထပ်ခါထပ်ခါ run ဖို့အတွက် အသုံးပြုပါတယ်။ Loops နှစ်မျိုးရှိပါတယ် - for loop နဲ့ while loop။ ဒီ guide မှာ loops တွေကို အသေးစိတ်ရှင်းပြပြီး example code တွေနဲ့ပြသပေးမှာဖြစ်ပါတယ်။

၅.၁။ for Loop

for loop ကို sequence (list, tuple, string, range, etc.) တစ်ခုထဲက items တွေကို iterate လုပ်ဖို့အတွက် အသုံးပြုပါတယ်။

Syntax:

```
for item in sequence:
    # Code to execute for each item
```

Example 1: List ထဲက items တွေကို iterate လုပ်ခြင်း

Code:

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Output:

```
apple
banana
cherry
```

Example 2: range() ကိုသုံးပြီး numbers တွေကို iterate လုပ်ခြင်း



Code:

```
for i in range(5): # 0 to 4
    print(i)
```

Output:

Result:

```
0
1
2
3
4
```

၅.၂။ while Loop

while loop က condition မှန်နေသမျှ code block ကို ထပ်ခါထပ်ခါ run ပါတယ်။

Syntax:

```
while condition:
    # Code to execute while condition is True
```

Example 3: while Loop

Code:

```
count = 0
while count < 5:

    print(count)
    count += 1
```



Output:

```
0
1
2
3
4
```

၅.၃။ Loop Control Statements

Loops ထဲမှာ flow ကို ထိန်းချုပ်ဖို့ control statements တွေကိုသုံးပါတယ်။

1. break

Loop ကို ရပ်တန့်ဖို့အတွက် break ကိုသုံးပါတယ်။

Example 4: break Statement

Code:

```
for i in range(10):
    if i == 5:
        break
    print(i)
```

Output:



Result:

0
1
2
3
4

2. continue

Loop ရဲ့ လက်ရှိ iteration ကို ကျော်ပြီး နောက် iteration ကို ဆက်သွားဖို့ continue ကိုသုံးပါတယ်။

Example 5: continue Statement

Code:

```
for i in range(5):  
    if i == 2:  
        continue  
    print(i)
```

Output:





0

1

3

4

3. pass

Code block ထဲမှာ ဘာမှမလုပ်ချင်ရင် pass ကိုသုံးပါတယ်။

Example 6: pass Statement

Code:

```
for i in range(3):  
    if i == 1:  
        pass # Do nothing  
    print(i)
```

Output:

0

1

2

၅.၄။ Nested Loops

Loop ထဲမှာ နောက်ထပ် loop တစ်ခုထပ်သုံးနိုင်ပါတယ်။

Example 7: Nested for Loop

Code:

```
for i in range(3):  
    for j in range(2):  
        print(f"i={i}, j={j}")
```

Output:

```
i=0, j=0  
i=0, j=1  
i=1, j=0  
i=1, j=1  
i=2, j=0  
i=2, j=1
```

၅.၅။ else in Loops

Loops မှာ else block ကို ထည့်နိုင်ပါတယ်။ Loop ပြီးသွားရင် else block ကို run ပါတယ်။ break နဲ့ loop ကိုရပ်ရင် else block ကို run မှာမဟုတ်ပါ။

Example 8: else in for Loop



Code:

```
for i in range(3):  
    print(i)  
else:  
    print("Loop finished!")
```

Output:

```
0  
1  
2  
Loop finished!
```

Example 9: else in while Loop

Code:

```
count = 0  
while count < 3:  
    print(count)  
    count += 1  
else:  
    print("Loop finished!")
```

Output:

```
0  
1  
2  
Loop finished!
```

၅.၆။ List Comprehensions



for loop ကို တိုတောင်းစွာရေးဖို့ List Comprehensions ကိုသုံးပါတယ်။

Syntax:

```
[expression for item in sequence]
```

Example 10: List Comprehension

Code:

```
squares = [x**2 for x in range(5)]  
print(squares)
```

Output:

```
[0, 1, 4, 9, 16]
```

၅.၇။ Best Practices

1. **Avoid Infinite Loops:** while loop မှာ condition ကို သတိထားပါ။
2. **Use break and continue Wisely:** Code readability ကို ထိခိုက်မှုမရှိအောင်သုံးပါ။
3. **List Comprehensions:** Simple loops တွေအတွက် List Comprehensions ကိုသုံးပါ။



✍ Short Notes:

1. for Loop: Sequence တစ်ခုထဲက items တွေကို iterate လုပ်ပါ။
2. while Loop: Condition မှန်နေသမျှ code block ကို run ပါ။
3. Control Statements: break, continue, pass တို့ကို loop flow ကို ထိန်းချုပ်ဖို့သုံးပါ။
4. Nested Loops: Loop ထဲမှာ နောက်ထပ် loop တစ်ခုထပ်သုံးပါ။
5. List Comprehensions: for loop တွေကို တိုတောင်းစွာရေးပါ။

Python Loops ကို ဒီနည်းနဲ့ လက်တွေ့အသုံးချနိုင်ပါတယ်။ Program logic တွေကို ပိုမိုတိကျစွာရေးသားနိုင်ပါတယ်။



လေ့ကျင့်ရန် -

Practice 1:

```
for i in range(5):
    print('loop no=',i)
```

Output:

```
Loop no= 0
Loop no= 1
Loop no= 2
Loop no= 3
Loop no= 4
```

Practice 2:

```
j=0  
while j<5:  
    print('loop no=',j)  
    j+=1 # j=j+1
```

Output:

```
Loop no= 0  
Loop no= 1  
Loop no= 2  
Loop no= 3  
Loop no= 4
```

Practice 3:

```
for row in range(3):  
    for col in range(3):  
        print('#')  
    print() #new line (break line)
```



Output:

```
#  
#  
#  
  
#  
#  
#  
  
#  
#  
#
```

Practice 4:

```
for row in range(5):  
    for col in range(10):  
        print('*',end='')  
    print() #new line (break line)
```

Output:

```
*****  
*****  
*****  
*****  
*****
```



Practice 5:

```
for row in range(7):  
    for col in range(10):  
        if row==0 or row==6:  
            print('*',end='')  
  
        if row>0 and row<6:  
            if col==0 or col==9:  
                print('*',end='')  
            else:  
                print(' ',end='')  
  
    print()
```

Output:

```
*****  
*           *  
*           *  
*           *  
*           *  
*           *  
*****
```

Practice 6:

```
for row in range(10):
    for col in range(10):
        if row==0 or row==4 or row==9:
            print('*',end='')
        if row>0 and row<4 or row>4 and row<9:
            if col==0 or col==9:
                print('*',end='')
            else:
                print(' ',end='')

    print()
```

Output:

```
*****
*           *
*           *
*           *
*****
*           *
*           *
*           *
*           *
*****
```




Practice 7:

```

for row in range(10):
    for col in range(20):
        if row == 0 or row == 4 or row == 9:
            print('*', end='')

        if row > 0 and row < 4:
            if col == 0 or col == 4 or col == 19:
                print('*', end='')
            else:
                print(' ', end='')

        if row > 4 and row < 9:
            if col == 0 or col == 19:
                print('*', end='')
            else:
                print(' ', end='')
    print()

```

Output:

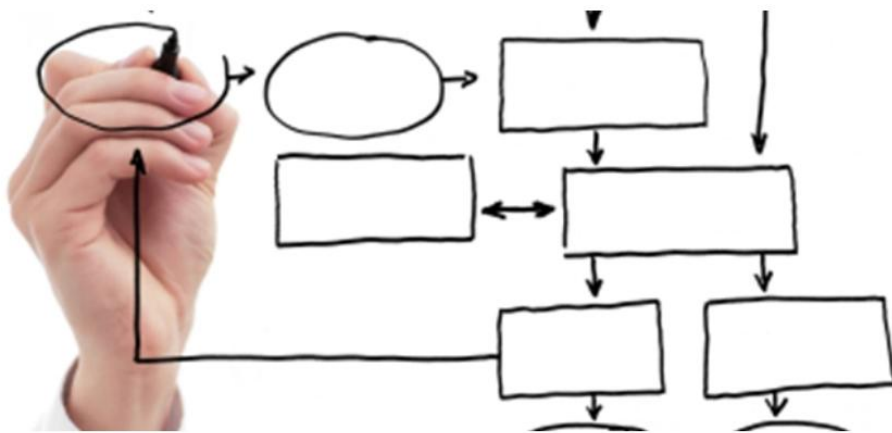
```

*****
*   *               *
*   *               *
*   *               *
*****
*                   *
*                   *
*                   *
*                   *
*****

```

အခန်း(၆)

Method



❑ Method

Method ဆိုသည်မှာ class အတွင်းတွင် သတ်မှတ်ထားသော function တစ်ခုဖြစ်ပြီး object များနှင့် သက်ဆိုင်သော အလုပ်များကို လုပ်ဆောင်ရန် အသုံးပြုပါသည်။ Method များကို အမျိုးအစား (၃) မျိုး ခွဲခြားနိုင်ပါသည်။

၆.၁။ Instance Methods

- **အဓိပ္ပာယ်:** Object တစ်ခု၏ attributes များကို လက်ရှိ instance နှင့် သက်ဆိုင်စွာ လုပ်ဆောင်သည်။
- **Syntax:** self parameter ကို ပထမဆုံး parameter အဖြစ် သတ်မှတ်ရပါမည်။
- **ဥပမာ:**

Code:

```
class Dog:
    def __init__(self, name):
        self.name = name # Instance attribute

    # Instance method
    def bark(self):
        print(f"{self.name} says Woof!")

# Object ဖန်တီးပြီး method ကို ခေါ်
my_dog = Dog("Buddy")
my_dog.bark() # Output: Buddy says Woof!
```

၆.၂။ Class Methods

- **အဓိပ္ပာယ်:** Class level တွင် လုပ်ဆောင်ပြီး class attribute များကို ပြင်ဆင်ရန်။

- **Syntax:** @classmethod decorator သုံးပြီး cls parameter ကို ပထမဆုံး parameter အဖြစ် သတ်မှတ်ရပါမည်။
- **ဥပမာ:**

Code:

```
class MyClass:
    count = 0 # Class attribute

    @classmethod
    def increment_count(cls):
        cls.count += 1 # Class attribute ကို ပြင်ဆင်

# Class method ကို ခေါ်
MyClass.increment_count()
print(MyClass.count) # Output: 1
```

၆.၃။ Static Methods

- **အဓိပ္ပာယ်:** Instance သို့မဟုတ် class နှင့် မသက်ဆိုင်သော function များအတွက်။
- **Syntax:** @staticmethod decorator သုံးပြီး self သို့မဟုတ် cls parameter မလိုအပ်ပါ။
- **ဥပမာ:**

Code:

```
class MathUtils:
    @staticmethod
    def add(a, b):
        return a + b # ရှိရှင်းသော ဂဏန်းပေါင်းခြင်း

# Static method ကို ခေါ်
print(MathUtils.add(5, 3)) # Output: 8
```

၆.၄။ Special/Magic Methods

- **အဓိပ္ပာယ်:** Python built-in functions များနှင့် သဟဇာတဖြစ်စေရန် သတ်မှတ်ထားသော methods များ။
- **Syntax:** `__method__` ပုံစံဖြင့် သတ်မှတ်ပါ။
- **ဥပမာ:**

Code:

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    # Object ကို string အဖြစ် ပြန်ပေးသည်
    def __str__(self):
        return f'{self.title} by {self.author}'

my_book = Book("Python 101", "John Doe")
print(my_book) # Output: 'Python 101' by John Doe
```

၆.၅။ Method Overriding (Inheritance)

- **အဓိပ္ပာယ်:** Parent class မှ method ကို child class တွင် အသစ်ပြန်သတ်မှတ်ခြင်း။
- **ဥပမာ:**

Code:

```
class Animal:
    def speak(self):
        print("Animal sound")

class Cat(Animal):
    def speak(self): # Method overriding
        print("Meow")
```



```
cat = Cat()
cat.speak() # Output: Meow
```

၆.၆။ Constructor Method (__init__)

- **အဓိပ္ပာယ်:** Object ဖန်တီးသည့်အခါ အလိုအလျောက် ခေါ်သော method ဖြစ်သည်။
- **ဥပမာ:**

Code:

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def display(self):
        print(f"{self.brand} {self.model}")

my_car = Car("Toyota", "Camry")
my_car.display() # Output: Toyota Camry
```

□ အဓိက ကွာခြားချက်များ

Method Type	Parameter	Decorator	Usage
Instance Method	self	None	Object attributes ကို ပြင်ဆင်
Class Method	cls	@classmethod	Class attributes ကို ပြင်ဆင်
Static Method	None	@staticmethod	သီးသန့် function လုပ်ဆောင်ခြင်း

✍ Short Notes:

1. **Instance Methods:** Object တစ်ခု၏ attributes များကို လုပ်ဆောင်ရာတွင် အသုံးပြုပါ။
2. **Class Methods:** Class level တွင် လုပ်ဆောင်ရာတွင် အသုံးပြုပါ။
3. **Static Methods:** လုပ်ဆောင်ချက်သည် class သို့မဟုတ် object နှင့် မသက်ဆိုင်ပါက အသုံးပြုပါ။
4. **Magic Methods:** Python built-in functions များနှင့် သဟဇာတဖြစ်စေရန် အသုံးပြုပါ။



လေ့ကျင့်ရန် -

Practice 1:

```
def exit():  
    print('Thanks for using my program ...')
```

```
def display(ans):  
    print('The answer is = ', ans)
```

```
def add_num(num1, num2):  
    return num1 + num2
```



```
def sub_num(num1, num2):  
    return num1 - num2  
  
def multiply_num(num1, num2):  
    return num1 * num2  
  
def divide_num(num1, num2):  
    return num1 / num2  
  
def get_input():  
    return int(input("Enter a number:"))  
  
def add():  
    x = get_input()  
    y = get_input()  
  
    ans = add_num(x, y)  
    display(ans)  
  
def sub():  
    x = get_input()  
    y = get_input()  
  
    ans = sub_num(x, y)  
    display(ans)  
  
def multiply():  
    x = get_input()  
    y = get_input()
```



```
ans = multiply_num(x, y)
display(ans)
```

```
def divide():
```

```
    x = get_input()
```

```
    y = get_input()
```

```
    ans = divide_num(x, y)
```

```
    display(ans)
```

```
def get_choice():
```

```
    return int(input('Enter Choice:'))
```

```
def menu():
```

```
    print(' Method Example 1  ')
```

```
    print(' ----- ')
```

```
    print(' [1] Add Numbers  ')
```

```
    print(' [2] Subtract Numbers  ')
```

```
    print(' [3] Multiple Numbers  ')
```

```
    print(' [4] Divide Numbers  ')
```

```
    print(' ----- ')
```

```
    print(' [5] Exit Program  ')
```

```
    print('-----')
```

```
def main():
```

```
    loop = True
```

```
    while loop:
```

```
        menu()
```



```
ch = get_choice()

if ch == 1:
    add()
elif ch == 2:
    sub()
elif ch == 3:
    multiply()
elif ch == 4:
    divide()
elif ch == 5:
    exit()
    loop = False
else:
    print('invalid choice...')
```

```
if __name__ == '__main__':
    main()
```

Output:

Method Example 1

[1] Add Numbers
[2] Subtract Numbers
[3] Multiply Numbers
[4] Divide Numbers

[5] Exit Program



Practice 2:

```
pi=3.14159
def exit():
    print('Thanks for using my program ...')
def back():
    print('Back to Main Menu...')

def display(ans):
    print('The answer is = ', ans)

def get_circle_area(r):
    return pi * r * r
def get_circle_circumference(r):
    return 2 * pi * r

def get_triangle_area(base, height):
    return base * height * 0.5

def get_triangle_volume(base, height):
    return (base * height) / 3

def circle_circumference():
    r = get_radius()

    ans = get_circle_circumference(r)
    display(ans)

def circle_area():
    r = get_radius()

    ans = get_circle_area(r)
    display(ans)
```



```
def circle():
    loop=1
    while loop:
        circle_menu()
        ch=get_choice()
        if ch==1:
            circle_area()
        elif ch==2:
            circle_circumference()
        elif ch==3:
            back()
        loop=0
    else:
        print('invlid choice...')
```

```
def triangle():
    loop=1
    while loop:
        triangle_menu()
        ch=get_choice()
        if ch==1:
            triangle_area()
        elif ch==2:
            triangle_volume()
        elif ch==3:
            back()
        loop=0
    else:
        print('invlid choice...')
```

```
def get_radius():
    return float(input('Enter Radius:'))
def get_base():
    return float(input('Enter Base:'))
```



```
def get_height():
    return float(input('Enter Height:'))

def triangle_area():
    b = get_base()
    h = get_height()

    ans = get_triangle_area(b, h)
    display(ans)
```

```
def triangle_volume():
    b = get_base()
    h = get_height()

    ans = get_triangle_volume(b, h)
    display(ans)
```

```
def get_choice():
    return int(input('Enter Choice:'))
```

```
def triangle_menu():
    print(' Triangle MENU  ')
    print(' ----- ')
    print(' [1] Triangle Area  ')
    print(' [2] Triangle Volume ')
    print('-----')
    print(' [3] Back  ')
    print('-----')
```

```
def circle_menu():
    print(' Circle MENU  ')
    print(' ----- ')
    print(' [1] Circle Area  ')
    print(' [2] Circle Circumference ')
    print('-----')
```



```
print(' [3] Back  ')\nprint('-----')\ndef main_menu():\n    print('  Method Example-2  ')\n    print(' ----- ')\n    print(' [1] Circle  ')\n    print(' [2] Triangle ')\n    print('-----')\n    print(' [3] Exit Program  ')\n    print('-----')
```

```
def main():\n    loop = 1\n    while loop:\n        main_menu()\n        ch = get_choice()\n\n        if ch == 1:\n            circle()\n\n        elif ch == 2:\n            triangle()\n\n        elif ch == 3:\n            exit()\n            loop = 0\n        else:\n            print('invalid choice...')
```

```
if __name__ == '__main__':\n    main()
```



Output:

Method Example -2

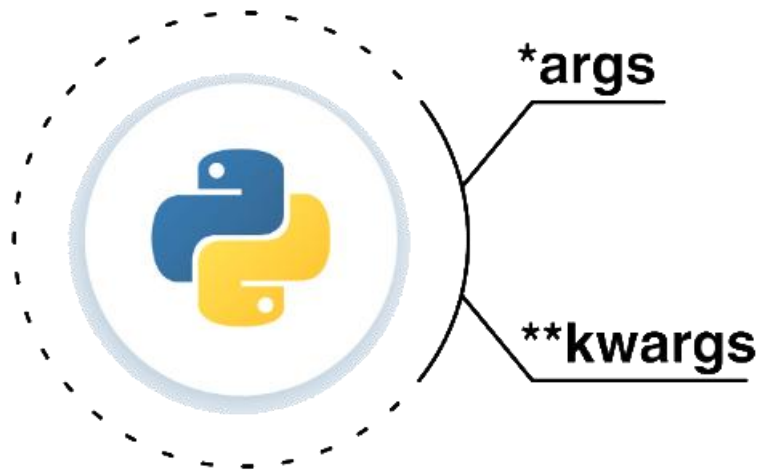
[1] Circle

[2] Triangle

[3] Exit Program

အခန်း(၇)

Arguments & Keyword Arguments



❑ Arguments & Keyword Arguments

Python မှာ function တွေကို ရေးတဲ့အခါ **arguments** နဲ့ **keyword arguments** တွေကို သုံးပြီး flexible ဖြစ်အောင် ရေးလို့ရပါတယ်။ ဒီနည်းလမ်းတွေကို သုံးပြီး function တွေကို ပိုပြီး dynamic ဖြစ်အောင် ရေးလို့ရပါတယ်။

၇.၁။ Arguments (Positional Arguments)

Arguments ဆိုတာက function ကို call လုပ်တဲ့အခါမှာ ပေးလိုက်တဲ့ values တွေကို ခေါ်ပါတယ်။ ဒီ values တွေကို function ရဲ့ parameters နဲ့ စဉ်လိုက် match လုပ်ပါတယ်။

Example: Basic Arguments

Code:

```
def greet(name, age):  
    print(f"Hello, {name}! You are {age} years old.")
```

```
greet("Mg Mg", 25)
```

Output:

```
Hello, Mg Mg! You are 25 years old.
```



ရှင်းလင်းချက်။

- name နဲ့ age ဆိုတဲ့ parameters နှစ်ခုကို function မှာ သတ်မှတ်ထားပါတယ်။
 - greet("Mg Mg", 25) လို့ call လုပ်တဲ့အခါ "Mg Mg" က name နဲ့ match ဖြစ်ပြီး 25 က age နဲ့ match ဖြစ်ပါတယ်။
-

၇.၂။ Keyword Arguments

Keyword arguments ကို သုံးပြီး arguments တွေကို parameter names နဲ့ တိုက်ရိုက် specify လုပ်လို့ရပါတယ်။ ဒီနည်းလမ်းက arguments တွေကို စဉ်လိုက် မပေးရပဲ လိုချင်တဲ့ parameter ကို တိုက်ရိုက် ပေးလို့ရပါတယ်။

Example: Keyword Arguments

Code:

```
def greet(name, age):
    print(f"Hello, {name}! You are {age} years old.")
```

```
greet(age=25, name="Mg Mg")
```

Output:

Hello, Mg Mg! You are 25 years old.

ရှင်းလင်းချက်။

- name="Mg Mg" နဲ့ age=25 ဆိုပြီး parameter names နဲ့ တိုက်ရိုက် values တွေကို ပေးထားပါတယ်။
- Arguments တွေကို စဉ်လိုက် မပေးရပဲ လိုချင်တဲ့ parameter ကို ရွေးပေးလို့ရပါတယ်။

၇.၃။ Default Arguments

Function မှာ parameters တွေကို default values တွေ သတ်မှတ်ထားလို့ရပါတယ်။ ဒါဆို function ကို call လုပ်တဲ့အခါမှာ ဒီ parameters တွေအတွက် values မပေးရင်လည်း default values တွေကို အလိုလို သုံးပါတယ်။

Example: Default Arguments

Code:

```
def greet(name, age=30):  
    print(f"Hello, {name}! You are {age} years old.")
```

```
greet("Mg Mg")  
greet("Bo Bo", 40)
```

Output:

```
Hello, Mg Mg! You are 30 years old.  
Hello, Bo Bo! You are 40 years old.
```

ရှင်းလင်းချက်။

- age parameter မှာ default value အနေနဲ့ 30 ကို သတ်မှတ်ထားပါတယ်။
 - greet("Mg Mg") လို့ call လုပ်တဲ့အခါ age အတွက် value မပေးလို့ default value 30 ကို အလိုလို သုံးပါတယ်။
 - greet("Bo Bo", 40) လို့ call လုပ်တဲ့အခါ age အတွက် value 40 ကို ပေးလိုက်ပါတယ်။
-

၇.၄။ Arbitrary Arguments (*args)

Function မှာ arguments အရေအတွက် မသေချာတဲ့အခါ *args ကို သုံးပါတယ်။ *args က arguments တွေကို tuple အနေနဲ့ လက်ခံပါတယ်။

Example: Arbitrary Arguments

Code:

```
def add_numbers(*args):  
    total = 0  
    for num in args:  
        total += num  
    return total
```

```
result = add_numbers(1, 2, 3, 4, 5)  
print(result) # Output: 15
```

ရှင်းလင်းချက်။

- *args ကို သုံးပြီး arguments တွေကို tuple အနေနဲ့ လက်ခံပါတယ်။
 - add_numbers(1, 2, 3, 4, 5) လို့ call လုပ်တဲ့အခါ args ထဲမှာ (1, 2, 3, 4, 5) ဆိုပြီး tuple အနေနဲ့ သိမ်းပါတယ်။
-

၇.၅။ Arbitrary Keyword Arguments (**kwargs)

Function မှာ keyword arguments အရေအတွက် မသေချာတဲ့အခါ **kwargs ကို သုံးပါတယ်။ **kwargs က keyword arguments တွေကို dictionary အနေနဲ့ လက်ခံပါတယ်။

Example: Arbitrary Keyword Arguments

Code:

```
def display_info(**kwargs):  
    for key, value in kwargs.items():  
        print(f"{key}: {value}")
```

```
display_info(name="Mg Mg", age=25, city="Mandalay")
```

Output:

```
name: Mg Mg  
age: 25  
city: Mandalay
```

ရှင်းလင်းချက်။

- `**kwargs` ကို သုံးပြီး keyword arguments တွေကို dictionary အနေနဲ့ လက်ခံပါတယ်။
- `display_info(name="Mg Mg", age=25, city="Mandalay")` လို့ call လုပ်တဲ့အခါ `kwargs` ထဲမှာ `{"name": "Mg Mg", "age": 25, "city": "Mandalay"}` ဆိုပြီး dictionary အနေနဲ့ သိမ်းပါတယ်။

၇.၆။ Combining Arguments and Keyword Arguments

Function တစ်ခုမှာ arguments, keyword arguments, `*args`, နဲ့ `**kwargs` တွေကို ပေါင်းစပ်ပြီး သုံးလို့ရပါတယ်။

Example:

Code:

```
def example_function(a, b, *args, x=10, y=20, **kwargs):  
    print(f"a: {a}, b: {b}")  
    print(f"args: {args}")  
    print(f"x: {x}, y: {y}")  
    print(f"kwargs: {kwargs}")
```

```
example_function(1, 2, 3, 4, x=15, name="Mg Mg", age=25)
```

Output:

```
a: 1, b: 2  
args: (3, 4)  
x: 15, y: 20  
kwargs: {'name': 'Mg Mg', 'age': 25}
```

ရှင်းလင်းချက်။

- a နဲ့ b က positional arguments တွေဖြစ်ပါတယ်။
- *args က additional positional arguments တွေကို tuple အနေနဲ့ လက်ခံပါတယ်။
- x နဲ့ y က keyword arguments တွေဖြစ်ပါတယ်။
- **kwargs က additional keyword arguments တွေကို dictionary အနေနဲ့ လက်ခံပါတယ်။

✍ Short Notes:

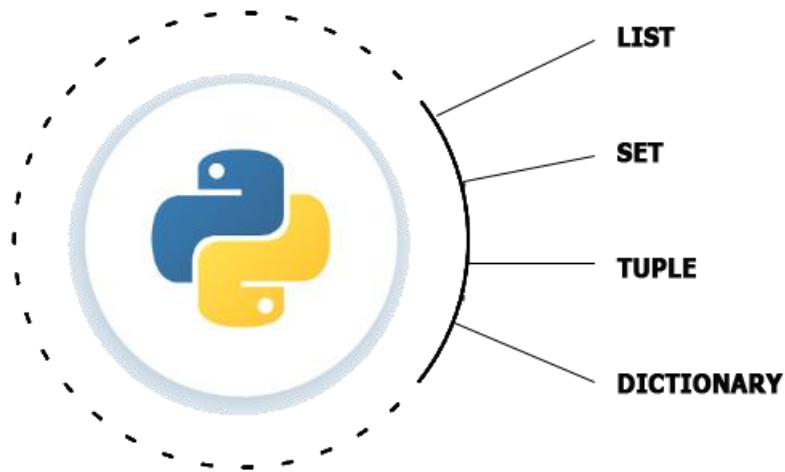
- Arguments: Function ကို call လုပ်တဲ့အခါ ပေးလိုက်တဲ့ values တွေကို positional အနေနဲ့ လက်ခံပါတယ်။
- Keyword Arguments: Parameter names နဲ့ တိုက်ရိုက် values တွေကို ပေးလိုရပါတယ်။
- Default Arguments: Parameters တွေကို default values တွေ သတ်မှတ်ထားလို့ရပါတယ်။
- *args: Additional positional arguments တွေကို tuple အနေနဲ့ လက်ခံပါတယ်။
- **kwargs: Additional keyword arguments တွေကို dictionary အနေနဲ့ လက်ခံပါတယ်။

ဒီ concepts တွေကို သေချာနားလည်ထားရင် Python functions တွေကို ပိုပြီး flexible ဖြစ်အောင် ရေးလို့ရပါတယ်။



အခန်း(၈)

Built-in Data Structures



❑ Built-in Data Structures:

- Lists
- Sets
- Tuples
- Dictionaries

Built-in data structure အမျိုးအစားများဖြစ်သော **Lists, Sets, Tuples, Dictionaries** တို့ကို အသုံးပြုပုံနှင့် သူတို့၏ ထူးခြားချက်များကို အသေးစိတ်ရှင်းပြပါမယ်။

၈.၁။ Lists (စာရင်းများ)

- **အဓိကလက္ခဏာများ:**
 - Ordered: အစဉ်လိုက် စီထားသော ဒေတာအစုအဝေး။
 - Mutable: ပြင်ဆင်နိုင်သည် (ဥပမာ - element များထည့်ခြင်း၊ ဖျက်ခြင်း)။
 - Duplicates Allowed: တူညီသော element များ ပါဝင်နိုင်တယ်။
 - Syntax: ထောင့်ကွင်းများ [] ဖြင့် ဖန်တီးပါတယ်။

- **ဥပမာ:**

Code:

```
my_list = [1, 2, 3, "apple", True]
```

- **အသုံးပြုနည်းများ:**

- **Element ထည့်ခြင်း:**

Code:

```
my_list.append(4)      # နောက်ဆုံးတွင် ထည့်သည် → [1, 2, 3, "apple", True, 4]
my_list.insert(1, "banana") # index 1 တွင် ထည့်သည် → [1, "banana", 2, 3, "apple", True]
```

- Element ဖျက်ခြင်း:

Code:

```
my_list.remove("apple") # တန်ဖိုးဖြင့် ဖျက်ခြင်း → [1, "banana", 2, 3, True]
popped = my_list.pop(2) # index 2 ကို ဖျက်ပြီး return ပြန် → popped = 2, my_list
= [1, "banana", 3, True]
```

- အခြား Method များ:

Code:

```
my_list.sort() # စီခြင်း (တူညီသော data type များသာ)
my_list.reverse() # ပြောင်းပြန် စီခြင်း
```



လေ့ကျင့်ရန် -

Practice 1:

```
number_list=[2,1,23,45,90]

print("list positional values...")
print(" index 0 =",number_list[0])
print(" index 1 =",number_list[1])
print(" index 2 =",number_list[2])

print("=====")
print("generate list values using loop index")
```

```
for i in range(len(number_list)):
    print(number_list[i])

print("=====")
print("generate list values using list content")

for num in number_list:
    print(num)
```

Output:

```
List positional values...
Index 0 =2
Index 1 = 1
Index 2= 23
=====
Generate list values using loop index
2
1
23
45
90
=====
Generate list values using list content
2
1
23
45
90
```



Practice 2:

```
student_list=["mg mg","aung aung","su su","tun tun"]

#display all studnets
print("Display all student list")
for student in student_list:
    print(student)

#add new student

new_student="bo bo"
student_list.append(new_student)

#display all studnets
print("Display all student list after adding new student bo bo")
for student in student_list:
    print(student)

#edit student name

#remove current data
student_list.remove(student_list[4])
#update new data
update_student="bo bo aung"
student_list.insert(4,update_student)
```



```
# display all studnets
print("Display all student list after updating student bo bo to bo bo aung")
for student in student_list:
    print(student)

# delete data
print("delete last student")
student_list.pop()

# display all studnets
print("Display all student list after deleting last index")
for student in student_list:
    print(student)

# delete data
print("delete student using del command")
del student_list[1]

# display all students
print("Display all student list after deleting an item using del keyword")
for student in student_list:
    print(student)
```

Output:

```
Display all student list
mg mg
aung aung
su su
tun tun
Display all student list after adding new student bo bo
mg mg
aung aung
su su
tun tun
bo bo
Display all student list after updating student bo bo to bo bo aung
mg mg
aung aung
su su
tun tun
bo bo aung
delete last student
Display all student list after deleting last index
mg mg
aung aung
su su
tun tun
delete student using del command
Display all student list after deleting an item using del keyword
mg mg
su su
tun tun
```



Practice 3:

```
#initialization of lists
contacts=[]

def exit_program():
    print('Exit Program Now...Bye Bye...')

def view_contacts():
    print( 'View Contacts' )
    for contact in contacts:
        temp=str(contact).split('-')
        print('-----')
        print('Name : ',temp[0])
        print('Address :',temp[1])
        print('-----')

def get_name():
    return input('Enter New Name:')

def get_address():
    return input('Enter New Address:')

def add_new_contact():
    print( 'Add New Contact' )
    new_name=get_name()
    new_address=get_address()

    new_contact=new_name+"-"+new_address
```



```
contacts.append(new_contact)

print('Saving Success')

def get_choice():
    return int(input('Enter Choice:'))

def main_menu():
    print('  Contact Menu  ')
    print('-----')
    print('[1] Add New Contact  ')
    print('[2] View Contacts  ')
    print('-----')
    print('[3] Exit  ')
    print('-----')

def main():
    loop=1
    while loop:
        main_menu()

        ch=get_choice()
        if ch==1:
            add_new_contact()
        elif ch==2:
            view_contacts()
        elif ch==3:
            exit_program()
        loop=0
    else:
        print('invalid choice...')
```




```
if __name__ == '__main__':  
    main()
```

Output:

Contact Menu

[1] Add New Contact

[2] View Contacts

[3] Exit

Add new Contact

Enter New Name: mg mg

Enter New Address: Hledan

View Contacts

Name: mg mg

Address: Hledan

၈.၂။ Tuples (ကျပ်ပယ်များ)

- **အဓိကလက္ခဏာများ:**

- Ordered: အစဉ်လိုက် စီထားသော ဒေတာအစုအဝေး။
- Immutable: ပြင်ဆင်၍မရပါ (element များကို ပြောင်းလဲ၍မရ)။
- Duplicates Allowed: တူညီသော element များ ပါဝင်နိုင်တယ်။
- Syntax: လက်သည်းကွင်းများ () ဖြင့် ဖန်တီးပါတယ်။

- **ဥပမာ:**

Code:

```
my_tuple = (1, 2, 3, "apple", True)
```

- **အသုံးပြုနည်းများ:**

- **Element ကို Access လုပ်ခြင်း:**

Code:

```
print(my_tuple[0]) # Output: 1
```

- **Immutable ဖြစ်သောကြောင့် ပြင်ဆင်၍မရ:**

Code:

```
my_tuple[0] = 5 # Error: 'tuple' object does not support item assignment
```

- **အခြား Method များ:**

Code:

```
index = my_tuple.index("apple") # "apple" ၏ index ကို ရှာခြင်း
```

```
count = my_tuple.count(2) # 2 ပါသည့် အကြိမ်ရေ
```



လေ့ကျင့်ရန် -

Practice 1:

```
student1=(1,'mg mg',22,'heldan')
student2=(2,'su su',22,'heldan')
print(student1)
```

```
print("id =",student1[0])
print("name =",student1[1])
print("age =",student1[2])
```

```
students=[]
```

```
students.append(student1)
students.append(student2)
```

```
print(students)
```

```
my_student_array=[]
```

```
my_student_array[1]=student1
my_student_array[2]=student2
```

```
print(my_student_array)
```



```
print(my_student_array[1])  
print(my_student_array[2])
```

Output:

```
(1, 'mg mg' , 22, 'hledan' )  
Id=1  
Name=mg mg  
Age=22  
[(1,' mg mg' ,22,' hledan' ),(2,' su su' ,22,' hledan' )]  
{1: ( 1,' mg mg' , 22, 'hledan' ), 2: (2,' su su' , 22, 'hledan' )}  
(1, 'mg mg' , 22, ' hledan' )  
(2,' su su' , 22, 'hledan' )
```



Practice 2:

```
class Account():
    def __init__(self,id,name,nrc,amount):
        self.id=id
        self.name=name
        self.nrc=nrc
        self.amount=amount

    def set_id(self,id):
        self.id=id
    def get_id(self):
        return self.id

    def set_name(self,name):
        self.name=name

    def get_name(self):
        return self.name

    def set_nrc(self,nrc):
        self.nrc=nrc
    def get_nrc(self):
        return self.nrc
    def set_amount(self,amount):
        self.amount=amount

    def get_amount(self):
        return self.amount
```

```
def display(self):
    print('-----')
    print('Id =',self.id)
    print('Name = ',self.name)
    print('Nrc =',self.nrc)
    print('Amount = ',self.amount)
    print('-----')
```

YomaBankApp.py

```
accounts=[]

def get_withdraw_amt():
    return int(input('Enter Withdraw amount:'))
def get_deposit():
    return int(input('Enter deposit amount:'))

def close_account():
    print('Close Account...')

found=0
found_index=-1
i=0

deposit_id=get_id()
for account in accounts:
    if deposit_id==account[0]:
        found=1
        found_index=i
        break
```

```
i+=1

if found==1:
    del accounts[found_index]
    print('account successfully closed...')
else:
    print('Sorry..account id not found..try again...')

def withdraw_fund():
    print('Deposit Fund...')

    found=0
    found_index=-1
    i=0

    deposit_id=get_id()
    for account in accounts:
        if deposit_id==account[0]:
            found=1
            found_index=i
            break
    i+=1

    if found==1:
        print('Account Found...')
        #since the object found is tuple we cannot change the value
        #that is why we need to convert it into list as the following
        curr_account=list(accounts[found_index])
        print('Current Amount is = ',curr_account[3])
        curr_amount=curr_account[3]

        withdraw_amt=get_withdraw_amt()

        if withdraw_amt> curr_amount:
            print('Sorry...Insufficient fund...try again...')
```

```
    print('Your current amount is ony...',curr_amount)
else:
    update_amount=curr_amount - deposit_amt

update_account=(curr_account[0],curr_account[1],curr_account[2],update_amount)
    del accounts[found_index]
    accounts.insert(found_index,update_account)

    print('Deposit Success...')

else:
    print('Sorry..Account Id not found...try agian...')

def deposit():
    print('Deposit Fund...')

    found=0
    found_index=-1
    i=0

    deposit_id=get_id()
    for account in accounts:
        if deposit_id==account[0]:
            found=1
            found_index=i
            break
        i+=1

    if found==1:
        print('Account Found...')
        #since the object found is tuple we cannot change the value
        #that is why we need to convert it into list as the following
        curr_account=list(accounts[found_index])
        print('Current Amount is = ',curr_account[3])
```



```
curr_amount=curr_account[3]
deposit_amt=get_deposit()
update_amount=curr_amount + deposit_amt
update_account=(curr_account[0],curr_account[1],curr_account[2],update_amount)
del accounts[found_index]
accounts.insert(found_index,update_account)
```

```
print('Deposit Success...')
```

```
else:
```

```
print('Sorry..Account Id not found...try again...')
```

```
def display(acc):
```

```
    print('-----')
```

```
    print('Account Id = ',acc[0])
```

```
    print('Account Name = ', acc[1])
```

```
    print('NRC = ',acc[2])
```

```
    print('Amount = ', acc[3])
```

```
    print('-----')
```

```
def exit_program():
```

```
    print('Exit Program Now..Bye Bye...')
```

```
def view_accounts():
```

```
    for account in accounts:
```

```
        display(account)
```

```
def get_id():
```

```
    return input('Enter Id:')
```

```
def get_name():
```

```
    return input('Enter Name:')
```

```
def get_nrc():
```

```
    return input('Enter Nrc:')
```



```
def get_amount():  
    return int(input('Enter Amount:'))
```

```
def create_account():  
    id=get_id()  
    name=get_name()  
    nrc=get_nrc()  
    amount=get_amount()  
  
    account=(id,name,nrc,amount)  
  
    accounts.append(account)  
  
    print('Saving Success...')
```

```
def get_choice():  
    return int(input('Enter Choice:'))  
  
def main_menu():  
    print(' Yoma Bank ')  
    print('-----')  
    print('  MENU  ')  
    print('-----')  
    print('[1] Create Account ')  
    print('[2] View Accounts ')  
  
    print('[3] Deposit Fund ')  
    print('[4] Widthdraw Fund ')  
    print('[5] Close Account ')  
    print('-----')  
    print('[6] Exit Program  ')  
    print('-----')
```



```
def main():  
    loop=1  
    while loop:  
        main_menu()  
        ch=get_choice()  
  
        if ch==1:  
            create_account()  
        elif ch==2:  
            view_accounts()  
        elif ch==3:  
            deposit_fund()  
        elif ch==4:  
            withdraw_fund()  
        elif ch==5:  
            close_account()  
        elif ch==6:  
            exit_program()  
        else:  
            print('invalid choice...')
```

```
if __name__=='__main__':  
    main()
```

Output:



Tuples Exercise

Yoma bank

MENU

[1] Create Account

[2] View Accounts

[3] Deposit Fund

[4] Withdraw Fund

[5] Close Account

[6] Exit Program

၈.၃။ Sets (အစုများ)

- **အဓိကလက္ခဏာများ:**

- **Unordered:** အစဉ်လိုက် မဟုတ်ပါ (index ဖြင့် access မလုပ်နိုင်)။
- **Mutable:** ပြင်ဆင်နိုင်သော်လည်း element များကို ပြောင်းလဲ၍မရ။
- **No Duplicates:** တူညီသော element များ မပါဝင်နိုင်။
- **Syntax:** ကွင်းနှစ် ခပ်များ ဖြင့် ဖန်တီးသည်။

- **ဥပမာ:**

Code:

```
my_set = {1, 2, 3, "apple"}
```

- **အသုံးပြုနည်းများ:**

- **Element ထည့်ခြင်း/ဖျက်ခြင်း:**

Code:

```
my_set.add("banana") # "banana" ထည့်ခြင်း
my_set.remove(2)     # 2 ကို ဖျက်ခြင်း
my_set.discard(5)     # 5 မရှိလျှင် error မထုတ်ပါ
```

- **Set Operations:**

Code:

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union = set1 | set2    # {1, 2, 3, 4, 5}
intersection = set1 & set2 # {3}
difference = set1 - set2 # {1, 2}
```



လေ့ကျင့်ရန် -

Practice 1:

```
fruits={"apple","banana","orange","mango"}
```

```
for fruit in fruits:
```

```
    print(fruit)
```

```
if "lime" not in fruits:
```

```
    print("No lime is not in fruit list")
```

```
else:
```

```
    print('yes...it is in the list')
```

```
fruits.add("lime")
```

```
print(fruits)
```

```
fruits.remove('apple')
```

```
print(fruits)
```

```
fruits.pop()
```



```
print(fruits)
fruits.discard('mango')
print(fruits)
del fruits
print(fruits)
```

output:

```
orange
mango
banana
apple
No lime is not in fruit list
{'orange', 'mango', 'banana', 'lime', 'apple'}
{'orange', 'mango', 'banana', 'lime'}
{'mango', 'banana', 'lime'}
{'banana', 'lime'}
{'banana', 'lime'}
```

Practice 2:

```
user_list=set()

message_list=[]

def chat_history():
    print('Chat History...')
    for message in message_list:
        print(message)
def send_message(sender,receiver,message):
    msg_obj={sender,receiver,message}
    message_list.append(msg_obj)

def get_message():
    return input('Enter Message:')

def check_receiver(sender,receiver):
    if sender!=receiver:
        if receiver in user_list:
            return True
        else:
            return False
    return False
def check_sender(sender):
    if sender in user_list:
        return True
```



```
return False
```

```
def chat():  
    print('Welcome to Simple Chat ')  
    sender=get_user()  
    receiver=get_user()  
  
    if check_sender(sender):  
        if check_receiver(sender,receiver):  
            message=get_message()  
            send_message(sender,receiver,message)  
            print('Successfully sent')  
        else:  
            print('sorry..receiver does not exist...')  
    else:  
        print('sorry..sender not found...')
```

```
def chat_history():  
    pass  
def exit_program():  
    print('Exit Program Now....')  
def view_users():  
    print('Active User List...')  
    for user in user_list:  
        print(user)  
def get_user():  
    return input("Enter new user name:")  
def add_user():
```

```
new_user=get_user()

user_list.add(new_user)

print('New User Added...')

def get_choice():
    return int(input('Enter choice:'))

def main_menu():
    print('  Set Example  ')
    print('-----')
    print('  Simple Chat  ')
    print('  Main MENU  ')
    print('-----')
    print('[1] Add new User')
    print('[2] View users ')
    print('[3] Chat (write message)  ')
    print('[4] Chat History  ')
    print('-----')
    print('[5] Exit  ')
    print('-----')

def main():
    loop=1

    while loop:

        main_menu()

        choice=get_choice()

        if choice==1:
```



```
        add_user()

    elif choice==2:

        view_users()

    elif choice==3:

        chat()

    elif choice==4:

        chat_history()

    elif choice==5:

        exit_program()

    else:

        print('Invalid choice...')

if __name__=='__main__':

    main()
```



output:

```
Set Example
```

```
-----
```

```
Main Menu
```

```
-----
```

```
[1] Add New User
```

```
[2] View Users
```

```
[3] Write Message
```

```
[4] View Messages
```

```
-----
```

```
[5] Exit
```

၈.၄။ Dictionaries (Key,Value)

- **အဓိကလက္ခဏာများ:**

- Unordered: Python 3.7+ တွင် insertion order ကို မှတ်သားထားသည်။
- Mutable: key-value pairs များကို ပြင်ဆင်နိုင်သည်။
- Keys are Unique: key များသည် ထူးခြားရပါမည်။
- Syntax: {key: value} ဖြင့် ဖန်တီးသည်။

- **ဥပမာ:**

Code:

```
my_dict = {"name": "Mg Mg", "age": 25, "city": "Yangon"}
```

- **အသုံးပြုနည်းများ:**



- **Element ကို Access လုပ်ခြင်း:**

Code:

```
print(my_dict["name"]) # Output: Mg Mg
print(my_dict.get("age")) # 25
```

- **Element ထည့်ခြင်း/ပြင်ဆင်ခြင်း:**

Code:

```
my_dict["email"] = "mgmg@example.com" # ထည့်ခြင်း
my_dict["age"] = 26 # ပြင်ဆင်ခြင်း
```

- **အခြား Method များ:**

Code:

```
keys = my_dict.keys() # ["name", "age", "city"]
values = my_dict.values() # ["Mg Mg", 25, "Yangon"]
items = my_dict.items() # [("name", "Mg Mg"), ("age", 25), ("city", "Yangon")]
```

□ အဓိက ကွာခြားချက်များ

Feature	List	Tuple	Set	Dictionary
Order	Ordered	Ordered	Unordered	Unordered (Python <3.7)
Mutability	Mutable	Immutable	Mutable	Mutable
Duplicates	Allowed	Allowed	Not Allowed	Keys are Unique
Syntax	[]	()	{}	{key: value}

❑ အသုံးပြုရန် သင့်တော်သော အခြေအနေများ

- **List:** အစဉ်လိုက် စီထားသော ဒေတာများကို လိုအပ်ပြီး ပြင်ဆင်ရန် လိုအပ်ပါက။
- **Tuple:** မပြောင်းလဲနိုင်သော ဒေတာများ (ဥပမာ - ကော်ဒီနိတ်များ)။
- **Set:** ထူးခြားသော element များနှင့် set operations လိုအပ်ပါက။
- **Dictionary:** Key-value pairs ဖြင့် ဒေတာများကို အမြန်ရှာဖွေရန် လိုအပ်ပါက။

ဥပမာ:

Code:

```
# List of student names
```

```
students = ["Mg Mg", "Su Su", "Aung Aung"]
```

```
# Tuple for coordinates
```

```
point = (10, 20)
```

```
# Set to track unique visitors
```

```
visitors = {"Mg Mg", "Bo Bo", "Mg Mg"} # {"Mg Mg", "Bo Bo"}
```

```
# Dictionary for user profile
```

```
user = {"name": "Mg Mg", "age": 25, "city": "Yangon"}
```



လေ့ကျင့်ရန် -

Practice 1:

```
product={'id':1,'name':'coke','price':1200}
```

```
print('product id =',product['id'])
```

```
print('product name =',product['name'])
```

```
print('product price =',product['price'])
```

```
product_list={1:{'id':1,'name':'coke','price':1200},2:{'id':2,'name':'apple','price':2000},3:{'id':3,'name':'pepsi','price':3000}}
```

```
print('retrieve object data by keys ')
```

```
print('product no 1 = ',product_list[1])
```

```
print('product no 2 = ',product_list[2])
```

```
print('retrieve object details ')
```

```
print('product no 1 = ',product_list[1]['name'])
```

```
print('product no 2 = ',product_list[3]['name'])
```

**Output:**

```
Product Id = 1
Product Name = coke
Product Price = 1200
Retrieve object data by keys
Product no 1 = { 'id' :1, 'name' : ' coke' , ' price' :1200}
Product no 2 = { 'id' :2, ' name' : ' apple' , ' price' :2000}
Retrieve object details
Product no 1 = coke
Product no 2 = pepsi
```

Practice 2:

Item.py

```
class item():
    #constructor
    def __init__(self,id,name,price):
        self.id=id
        self.name=name
        self.price=price
    #getters and setters
    def get_id(self):
        return self.id

    def set_id(self,id):
```




```
self.id=id

def get_name(self):
    return self.name

def set_name(self,name):
    self.name=name

def get_price(self):
    return self.price

def set_price(self,price):
    self.price=price

def display(self):
    print('-----')
    print('Id : ',self.id)
    print('Name : ',self.name)
    print('Price :',self.price)
    print('-----')
```

SeinGayHarApp.py

```
from Item import item

#dictionary declaration
item_list={}

def exit_program():
    print('Exit Program Now..Bye Bye...')
def view_items():
    for item in item_list.values():
```



```
        item.display()
def get_id():
    return input('Enter Id:')

def get_name():
    return input('Enter Name:')

def get_price():
    return input('Enter Price:')

def add_new_item():
    id=get_id()
    name=get_name()
    price=get_price()

    new_item=item(id,name,price)

    item_list[id]=new_item

    print('Saving Success...')

def get_choice():
    return int(input('Enter Choice:'))
def main_menu():
    print( '----- ' )
    print( ' Dictionary Practice 2' )
    print(' Sein Gay Har ' )
    print('-----')
    print('  MENU  ')
    print('-----')
    print('[1] Add New Item ')
    print('[2] Show Items ist ')
    print('-----')
```



```
print('[3] Exit Program  ')
print('-----')
```

```
def main():
    loop=1
    while loop:

        main_menu()
        ch=get_choice()

        if ch==1:
            add_new_item()
        elif ch==2:
            view_items()
        elif ch==3:
            exit_program()
            loop=0
        else:
            print('invalid choice...')

if __name__=='__main__':
    main()
```



Output:

Sein Gay Har

MENU

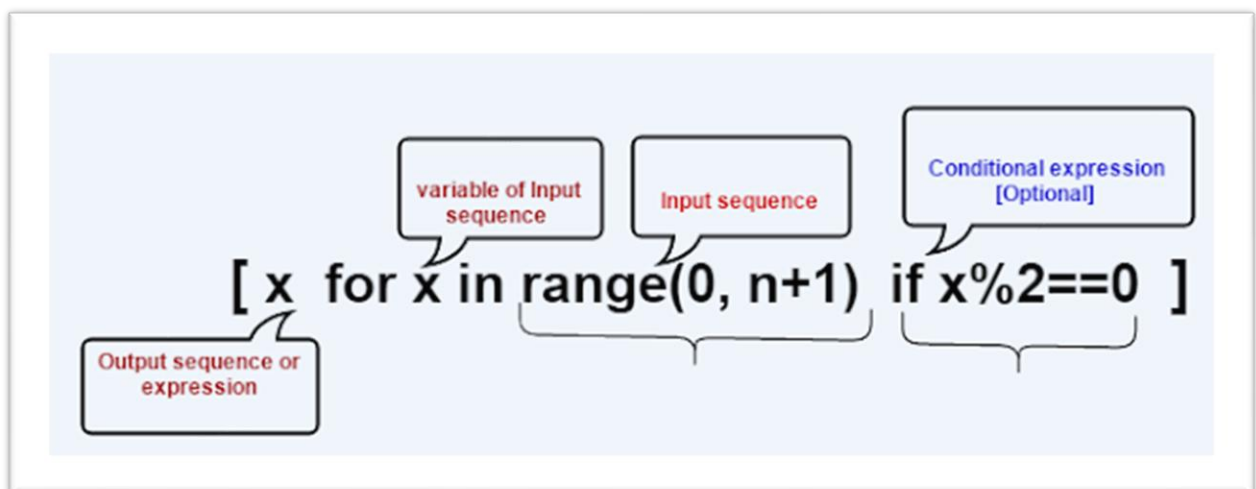
[1] Add New item

[2] Show Items List

[3] Exit Program

အခန်း(၉)

List Comprehensions



❑ List Comprehension

List Comprehension ဆိုတာက list တစ်ခုကို ရိုးရှင်းစွာနဲ့ လျင်မြန်စွာ ဖန်တီးနိုင်တဲ့ နည်းလမ်းတစ်ခုဖြစ်ပါတယ်။ List comprehension ကို သုံးပြီး loop တွေ၊ condition တွေနဲ့ list တစ်ခုကို တစ်ကြောင်းတည်းနဲ့ ဖန်တီးနိုင်ပါတယ်။ ဒီနည်းလမ်းက code ကို ပိုပြီး readable ဖြစ်စေပြီး concise ဖြစ်စေပါတယ်။

❑ List Comprehension Syntax

List comprehension ရဲ့အခြေခံ syntax က အောက်ပါအတိုင်းဖြစ်ပါတယ်။

Code:

[expression for item in iterable if condition]

- **expression:** List ထဲမှာ ထည့်မယ့် value ကို ဖော်ပြပါတယ်။
- **item:** Iterable (ဥပမာ list, tuple, string) ထဲက တစ်ခုချင်းစီကို ကိုယ်စားပြုပါတယ်။
- **iterable:** Loop ပတ်မယ့် data structure (ဥပမာ list, range, string)။
- **condition (optional):** Condition ကို စစ်ဆေးပြီး မှန်မှသာ list ထဲမှာ ထည့်ပါတယ်။

၉.၁။ Basic List Comprehension

ဥပမာ: 1 ကနေ 5 အထိ ဂဏန်းတွေကို list တစ်ခုအနေနဲ့ ဖန်တီးကြည့်ရအောင်။

Code:

```
# Without list comprehension
```

```
numbers = []
```

```
for i in range(1, 6):
```

```
    numbers.append(i)
```

```
print(numbers) # Output: [1, 2, 3, 4, 5]
```

```
# With list comprehension
numbers = [i for i in range(1, 6)]
print(numbers) # Output: [1, 2, 3, 4, 5]
```

- **Explanation:**

- i ကို range(1, 6) ထဲကနေ တစ်ခုချင်းစီယူပြီး list ထဲမှာ ထည့်ပါတယ်။
 - List comprehension ကို သုံးတာက ပိုပြီး concise ဖြစ်ပါတယ်။
-

၉.၂။ List Comprehension with Condition

ဥပမာ: 1 ကနေ 10 အထိ ဂဏန်းတွေထဲက စုံဂဏန်းတွေကိုပဲ list ထဲမှာ ထည့်ကြည့်ရအောင်။

Code:

```
# Without list comprehension
even_numbers = []
for i in range(1, 11):
    if i % 2 == 0:
        even_numbers.append(i)
print(even_numbers) # Output: [2, 4, 6, 8, 10]
```

```
# With list comprehension
even_numbers = [i for i in range(1, 11) if i % 2 == 0]
print(even_numbers) # Output: [2, 4, 6, 8, 10]
```

- **Explanation:**

- if i % 2 == 0 ဆိုတဲ့ condition ကို စစ်ဆေးပြီး မှန်မှသာ list ထဲမှာ ထည့်ပါတယ်။
-

၉.၃။ Nested List Comprehension

ဥပမာ: Nested loop တွေကို list comprehension နဲ့ ရေးကြည့်ရအောင်။

Code:

```
# Without list comprehension
```

```
matrix = []
for i in range(3):
    row = []
    for j in range(3):
        row.append(i + j)
    matrix.append(row)
print(matrix) # Output: [[0, 1, 2], [1, 2, 3], [2, 3, 4]]
```

With list comprehension

```
matrix = [[i + j for j in range(3)] for i in range(3)]
print(matrix) # Output: [[0, 1, 2], [1, 2, 3], [2, 3, 4]]
```

ရှင်းလင်းချက်။

- ပထမ loop (for i in range(3)) က outer list ကို ဖန်တီးပါတယ်။
 - ဒုတိယ loop (for j in range(3)) က inner list ကို ဖန်တီးပါတယ်။
-

၉.၄။ List Comprehension with Expressions

ဥပမာ: List ထဲက ဂဏန်းတွေကို square လုပ်ကြည့်ရအောင်။

Code:

Without list comprehension

```
squares = []
for i in range(1, 6):
    squares.append(i ** 2)
print(squares) # Output: [1, 4, 9, 16, 25]
```

With list comprehension

```
squares = [i ** 2 for i in range(1, 6)]
print(squares) # Output: [1, 4, 9, 16, 25]
```


ရှင်းလင်းချက်။

- `i ** 2` ဆိုတဲ့ expression ကို သုံးပြီး list ထဲမှာ ထည့်ပါတယ်။
-

၉.၅။ List Comprehension with Strings

ဥပမာ: String တစ်ခုထဲက စာလုံးတွေကို uppercase ပြောင်းပြီး list ထဲမှာ ထည့်ကြည့်ရအောင်။

Code:

```
# Without list comprehension
```

```
text = "hello"
```

```
letters = []
```

```
for char in text:
```

```
    letters.append(char.upper())
```

```
print(letters) # Output: ['H', 'E', 'L', 'L', 'O']
```

```
# With list comprehension
```

```
letters = [char.upper() for char in text]
```

```
print(letters) # Output: ['H', 'E', 'L', 'L', 'O']
```

ရှင်းလင်းချက်။

- `char.upper()` ဆိုတဲ့ expression ကို သုံးပြီး string ထဲက စာလုံးတွေကို uppercase ပြောင်းပါတယ်။
-

၉.၆။ List Comprehension with Multiple Conditions

ဥပမာ: 1 ကနေ 20 အထိ ဂဏန်းတွေထဲက 3 နဲ့ 5 နဲ့ စားလို့ပြတ်တဲ့ ဂဏန်းတွေကို ရှာကြည့်ရအောင်။

Code:

```
# Without list comprehension
numbers = []
for i in range(1, 21):
    if i % 3 == 0 and i % 5 == 0:
        numbers.append(i)
print(numbers) # Output: [15]
```

```
# With list comprehension
numbers = [i for i in range(1, 21) if i % 3 == 0 and i % 5 == 0]
print(numbers) # Output: [15]
```

ရှင်းလင်းချက်။

- if i % 3 == 0 and i % 5 == 0 ဆိုတဲ့ condition ကို စစ်ဆေးပါတယ်။

❑ Advantages of List Comprehension

1. **Concise:** Code ကို ပိုပြီး တိုတောင်းစေပါတယ်။
2. **Readable:** ရှိရင်းတဲ့ logic တွေကို ပိုပြီး ဖတ်ရလွယ်စေပါတယ်။
3. **Faster:** တခါတရံမှာ loop တွေထက် ပိုပြီး မြန်ဆန်ပါတယ်။



❑ When Not to Use List Comprehension

1. **Complex Logic:** Logic က ရှုပ်ထွေးနေရင် list comprehension ကို မသုံးသင့်ပါဘူး။
 2. **Readability:** Code ကို ဖတ်ရခက်သွားရင် ရိုးရိုး loop ကို ပြန်သုံးသင့်ပါတယ်။
-

✍ Short Notes:

List comprehension က Python မှာ အသုံးများတဲ့ feature တစ်ခုဖြစ်ပြီး၊ code တွေကို ပိုပြီး concise နဲ့ readable ဖြစ်အောင် ကူညီပေးပါတယ်။ ဒီနည်းလမ်းကို သေချာနားလည်ထားရင် Python programming မှာ ပိုပြီး effective ဖြစ်စေပါတယ်။



အခန်း(၁၀)

Lambda, Map and Filter



❑ Lambda, Map and Filter

Lambda, Map, နှင့် Filter တို့သည် functional programming အတွက် အသုံးဝင်သော features များဖြစ်ပါသည်။ သူတို့ရဲ့အသုံးပြုပုံနှင့် ဥပမာများကို အောက်တွင် ရှင်းပြထားပါတယ်။

၁၀.၁။ Lambda Functions

- Lambda သည် အမည်မဲ့ function ဖြစ်ပြီး ရိုးရှင်းသော operations များအတွက် သုံးပါတယ်။
- Syntax: lambda arguments: expression
- အသုံးပြုပုံ: လိုအပ်တဲ့ function ကို တစ်ကြောင်းတည်းဖြင့် ရေးနိုင်ပါတယ်။

Example 1: Basic Lambda

Code:

```
# Regular function
def add(a, b):
    return a + b

# Lambda equivalent
add_lambda = lambda a, b: a + b

print(add(2, 3))      # Output: 5
print(add_lambda(2, 3)) # Output: 5
```

Example 2: Lambda with Conditional

Code:

```
# Check even number
```

```
is_even = lambda x: True if x % 2 == 0 else False
```

```
print(is_even(4)) # Output: True
```

```
print(is_even(5)) # Output: False
```



လေ့ကျင့်ရန်-

Practice 1:

```
add = lambda a,b : a + b
```

```
sub = lambda a,b : a -b
```

```
mul = lambda a,b : a * b
```

```
div = lambda a,b : a/b
```

```
print(add(5,2))
```

```
print(sub(5,2))
```

```
print(mul(5,2))
```

```
print(div(5,2))
```



Output:

```
7
3
10
2.5
```

Practice 2:

```
def to_upper(str):
    return lambda str: str.upper()

upper=to_upper(str)
print(upper('Welcome to Northern City'))
```

Output:

```
WELCOME TO NORTHERN CITY
```



Practice 3:

```
List = [[2,3,4],[1, 4, 16, 64],[3, 6, 9, 12]]
```

```
sortList = lambda x: (sorted(i) for i in x)
```

```
largest = lambda x, f : [y[len(y)-1] for y in f(x)]
```

```
findLargest = largest(List, sortList)
```

```
print(findLargest)
```

Output:

```
[4, 64, 12]
```

၁၀.၂။ Map Function

- အဓိပ္ပာယ်: Iterable (list, tuple, etc.) ထဲက element တိုင်းကို function တစ်ခုနဲ့ process လုပ်ပြီး map object ပြန်ပေးပါတယ်။
- Syntax: map(function, iterable)
- အသုံးပြုပုံ: တူညီသော operation ကို element တိုင်းကို apply လုပ်ရန်။

Example 1: Square Numbers

Code:

```
numbers = [1, 2, 3, 4]

# Using map with lambda
squared = map(lambda x: x ** 2, numbers)
print(list(squared)) # Output: [1, 4, 9, 16]
```

Example 2: Convert to Uppercase

Code:

```
words = ["apple", "banana", "cherry"]

# Using map with lambda
uppercase = map(lambda word: word.upper(), words)
print(list(uppercase)) # Output: ['APPLE', 'BANANA', 'CHERRY']
```



လေ့ကျင့်ရန် -

Practice 1:

```
def myfunc(a, b):
    return a + ' '+b
first_names=["mg mg","bo bo","su myat"]
```



```
last_names=["kyaw","tun","mon"]
res = map(myfunc, first_names, last_names)

print(res)

#convert the map into a list, for readability:
print(list(res))
```

Output:

```
<map object at 0x14ec3931f2b0>
['mg mg kyaw', 'bo bo tun', 'su myat mon']
```

Practice 2:

```
def myMapFunc(n):
    return n.upper()

my_tuple = ('php','java','python','c++','c')

updated_list = map(myMapFunc, my_tuple)
print(updated_list)
print(list(updated_list))
```

Output:

```
<map object at 0x15094e2c4130>
['PHP', 'JAVA', 'PYTHON', 'C++', 'C']
```



Practice 3:

```
def mapping_course(courses, fees):
    return courses+" : "+fees
```

```
course_titles = ['Computer Basic','Advanced Excel', 'Graphic', 'Video Editing', 'UIUX Design']
```

```
course_fees = ('50,000 ks','60,000 ks','100,000 ks','120,000 ks','150,000 ks')
```

```
updated_list = map(mapping_course, course_titles, course_fees)
print(list(updated_list))
```

Output:

```
['Computer Basic : 50,000 ks', 'Advanced Excel : 60,000 ks', 'Graphic : 100,000 ks', 'Video Editing : 120,000 ks', 'UIUX Design : 150,000 ks']
```

၁၀.၃။ Filter Function

- အဓိပ္ပာယ်: Iterable ထဲက element တွေထဲမှ function က True ပြန်တဲ့ element တွေကို စစ်ထုတ်ပြီး filter object ပြန်ပေးပါတယ်။
- Syntax: filter(function, iterable)
- အသုံးပြုပုံ: Condition တစ်ခုနဲ့ ကိုက်ညီတဲ့ element တွေကို ရွေးရန်။

Example 1: Filter Even Numbers

Code:

```
numbers = [1, 2, 3, 4, 5, 6]

# Using filter with lambda
even_numbers = filter(lambda x: x % 2 == 0, numbers)
print(list(even_numbers)) # Output: [2, 4, 6]
```

Example 2: Filter Words Longer Than 5 Letters

Code:

```
words = ["apple", "banana", "kiwi", "orange"]

# Using filter with lambda
long_words = filter(lambda word: len(word) > 5, words)
print(list(long_words)) # Output: ['banana', 'orange']
```

၁၀.၄။ Lambda + Map + Filter တွဲသုံးခြင်း

Example: Even Numbers များကို Square လုပ်ခြင်း

Code:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8]

# Step 1: Filter even numbers
even_numbers = filter(lambda x: x % 2 == 0, numbers)

# Step 2: Square the even numbers
squared_evens = map(lambda x: x ** 2, even_numbers)

print(list(squared_evens)) # Output: [4, 16, 36, 64]
```



❑ အဓိက ကွာခြားချက်များ

Feature	Lambda	Map	Filter
အဓိကအလုပ်	ရိုးရှင်းသော function ဖန်တီးရန်	Iterable ထဲက element တိုင်းကို process လုပ်ရန်	Iterable ထဲက element များကို condition နဲ့စစ်ရန်
Return Type	Function object	Map object (iterator)	Filter object (iterator)
အသုံးပြုသည့်နေရာ	လိုအပ်သော function ကို တစ်ကြောင်းတည်းဖြင့်ရေးရန်	တူညီသော operation ကို element အားလုံးကို apply လုပ်ရန်	သတ်မှတ်ထားသော condition နှင့်ကိုက်ညီသော element များကိုရွေးရန်

✍ Short Notes

1. Lambda ကို ရိုးရှင်းသော operations များအတွက်သာ သုံးပါ (complex logic များအတွက် regular functions ကို သုံးပါ)။
2. Map နှင့် Filter ကို list comprehensions နှင့် နှိုင်းယှဉ်ပြီး ရွေးချယ်ပါ (list comprehensions က ပိုဖတ်ရလွယ်ပါတယ်)။
3. Map object နှင့် Filter object များကို list အဖြစ် ပြောင်းရန် list() ကို သုံးပါ။

ဥပမာ: List Comprehension vs Map+Lambda

Code:

```
# Using map + lambda
squared = map(lambda x: x ** 2, [1, 2, 3])
print(list(squared)) # Output: [1, 4, 9]
```

```
# Using list comprehension
squared = [x ** 2 for x in [1, 2, 3]]
print(squared) # Output: [1, 4, 9]
```



လေ့ကျင့်ရန် -

Practice 1:

```
items=[
    {"id":1,"brand":"dell","cpu":"corei3","ram":"8GB","price":400},
    {"id":2,"brand": "acer","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":3,"brand": "hp","cpu": "corei5", "ram": "8GB", "price": 600},
    {"id":4,"brand": "lenovo","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":5,"brand": "dell","cpu": "corei7", "ram": "16GB", "price": 800},
    {"id":6,"brand": "dell","cpu": "corei5", "ram": "16GB", "price": 500},
    {"id":7,"brand": "hp","cpu": "corei3", "ram": "16GB", "price": 700},
    {"id":8,"brand": "hp","cpu": "corei3", "ram": "8GB", "price": 500},
    {"id":9,"brand": "acer","cpu": "corei7", "ram": "32GB", "price": 900},
    {"id":10,"brand": "acer","cpu": "corei3", "ram": "8GB", "price": 500},
    {"id":11,"brand": "asus","cpu": "corei7", "ram": "16GB", "price": 800},
    {"id":12,"brand": "asus","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":13,"brand": "acer","cpu": "corei7", "ram": "32GB", "price":1200},
    {"id":14,"brand": "acer","cpu": "corei7", "ram": "16GB", "price": 500},
    {"id":15,"brand": "lenovo","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":16,"brand": "acer","cpu": "corei3", "ram": "16GB", "price": 500}
];
```



```
print('display dell laptop computer list')
```

```
dell_laptops=list(filter(lambda item:item['brand']=='dell',items))
```

```
for item in dell_laptops:  
    print(item)
```

```
print('display acer laptop computer list')
```

```
acer_laptops=list(filter(lambda item:item['brand']=='acer',items))
```

```
for item in acer_laptops:  
    print(item)
```

```
print('display laptop list which price is greater than 600 ')
```

```
morethan_600_laptops=list(filter(lambda item:item['price']>=600,items))
```

```
for item in morethan_600_laptops:  
    print(item)
```

```
print('display laptop cpu corei7 list ')
```

```
corei7_laptops=list(filter(lambda item:item['cpu']=='corei7',items))
```

```
for item in corei7_laptops:  
    print(item)
```

Output:

```
display dell laptop computer List
{"id":1,"brand":"dell","cpu":"corei3","ram":"8GB","price":400},
{"id":5,"brand": "dell", "cpu": "corei7", "ram": "16GB", "price": 800}
{"id":6,"brand": "dell", "cpu": "corei5", "ram": "16GB", "price": 500}
display acer laptop computer List
{"id":2,"brand": "acer ", "cpu": "corei3", "ram": "16GB", "price": 500},
{"id":9,"brand": "acer", "cpu": "corei7", "ram": "32GB", "price": 900}
{"id":10,"brand": "acer ", "cpu": "corei3", "ram": "8GB", "price": 500}
{"id":13,"brand": "acer ", "cpu": "corei7", "ram": "32GB", "price": 1200},
{"id":14,"brand": "acer", "cpu": "corei7", "ram": "16GB", "price": 500}
{"id":16,"brand": "acer ", "cpu": "corei3", "ram": "16GB", "price": 500}
display laptop List which price is greater than 600
{"id":3,"brand": "hp ", "cpu": "corei5", "ram": "16GB", "price": 600},
{"id":5,"brand": "dell", "cpu": "corei7", "ram": "32GB", "price": 800}
{"id":7,"brand": "hp ", "cpu": "corei3", "ram": "8GB", "price": 700}
{"id":9,"brand": "acer ", "cpu": "corei7", "ram": "32GB", "price": 900},
{"id":11,"brand": "asus", "cpu": "corei7", "ram": "16GB", "price": 800}
{"id":13,"brand": "acer ", "cpu": "corei7", "ram": "16GB", "price": 1200}
Display laptop cup corei7 list
{"id":5,"brand": "dell ", "cpu": "corei7", "ram": "16GB", "price": 800},
{"id":9,"brand": "acer", "cpu": "corei7", "ram": "32GB", "price": 900}
{"id":11,"brand": "asus ", "cpu": "corei7", "ram": "16GB", "price": 800}
{"id":13,"brand": "acer ", "cpu": "corei7", "ram": "32GB", "price": 1200},
{"id":14,"brand": "acer", "cpu": "corei7", "ram": "16GB", "price": 500}
```


အခန်း(၁၁)

Random Numbers & String





❑ Random Numbers & String

Python တွင် **Random Numbers** နှင့် **Random Strings** များ ထုတ်လုပ်နည်းကို လွယ်ကူစွာ နားလည်နိုင်ရန် Step-by-Step ရှင်းလင်းချက်များနှင့် ဥပမာများကို အောက်မှာ ဖော်ပြထားပါတယ်။

၁၁.၁။ Random Numbers ထုတ်လုပ်ခြင်း

Python ၏ random module ကို အသုံးပြု၍ random numbers များကို ထုတ်ယူနိုင်ပါတယ်။

1: random Module ကို Import လုပ်ပါ

Code:

```
import random
```

2: အခြေခံ Random Numbers များ

- **random.randint(a, b):** a နှင့် b ကြား random integer ထုတ်ပေးမယ် (a နှင့် b ပါဝင်)။

Code:

```
print(random.randint(1, 10)) # 1 မှ 10 ကြား random integer
```

- **random.random():** 0.0 မှ 1.0 ကြား random float ထုတ်ပေးသည်။

Code:

```
print(random.random()) # e.g., 0.5488135023
```

- **random.uniform(a, b):** a နှင့် b ကြား random float ထုတ်ပေးသည်။

Code:

```
print(random.uniform(5.5, 10.5)) # 5.5 မှ 10.5 ကြား float
```

3: Seed သတ်မှတ်ခြင်း (Reproducible Random Numbers)

Random numbers များကို ထပ်တူထုတ်လိုပါက `seed()` ကို သုံးပါ။

Code:

```
random.seed(42) # Seed ကို 42 ဟု သတ်မှတ်  
print(random.randint(1, 100)) # ထပ်ခါခါတိုင်း အလားတူအဖြေ
```

၁၁.၂။ Random Strings ထုတ်လုပ်ခြင်း

Random strings များကို `random module` နှင့် `string module` တို့ကို ပေါင်းစပ်အသုံးပြု၍ ထုတ်လုပ်နိုင်ပါသည်။

၁၁.၂.၁။ Modules များကို Import လုပ်ပါ

Code:

```
import random  
import string
```

၁၁.၂.၂။ စာလုံးများကို သတ်မှတ်ပါ

Code:

```
# စာလုံးများ (အကြီး + အသေး)  
letters = string.ascii_letters  
# ဂဏန်းများ  
digits = string.digits  
# အထူးသင်္ကေတများ  
symbols = string.punctuation
```

၁၁.၂.၃။ Random String ထုတ်လုပ်ခြင်း

- `random.choices()` ကို အသုံးပြုခြင်း (အက္ခရာများ ထပ်နိုင်တယ်)။

Code:

```
all_chars = letters + digits + symbols
random_string = ''.join(random.choices(all_chars, k=10))
print(random_string) # e.g., "aB3$kL9@mN"
```

- **random.sample() ကို အသုံးပြုခြင်း** (အကွဲပြားမှုများ မထပ်ရ)။

Code:

```
unique_chars = random.sample(all_chars, k=10)
random_string = ''.join(unique_chars)
print(random_string) # e.g., "5tG$fKpL2d"
```

၁၁.၃။ လုံခြုံသော Random Strings (ဥပမာ - Passwords)

လုံခြုံရေးအတွက် secrets module ကို အသုံးပြုပါ။

Code:

```
import secrets
import string

# စာလုံးများကို သတ်မှတ်ပါ
characters = string.ascii_letters + string.digits + string.punctuation

# လုံခြုံသော password ထုတ်လုပ်ခြင်း
secure_password = ''.join(secrets.choice(characters) for _ in range(12))
print(secure_password) # e.g., "s9F@kLm$5zTq"

# URL-safe token ထုတ်လုပ်ခြင်း
url_safe_token = secrets.token_urlsafe(16)
print(url_safe_token) # e.g., "Dwk3aXQ7eBZcNIO9fjK-2g"
```

၁၁.၄။ အခြား အသုံးဝင်သော Functions

- **Random Hexadecimal String:**

Code:

```
hex_token = secrets.token_hex(8) # 8 bytes → 16 characters
print(hex_token) # e.g., "1a3f5c7e9b2d4f6a"
```

- **Random UUID:**

Code:

```
import uuid
unique_id = uuid.uuid4()
print(unique_id) # e.g., "f47ac10b-58cc-4372-a567-0e02b2c3d479"
```

❑ ဥပမာများ

Example 1: Random OTP (6 Digits)

Code:

```
import random
otp = random.randint(100000, 999999)
print(f"Your OTP is: {otp}") # e.g., 348572
```

Example 2: Random Username (8 Characters)

Code:

```
import random
import string

username = ''.join(random.choices(string.ascii_lowercase + string.digits, k=8))
print(username) # e.g., "alice123"
```



Example 3: Secure API Key

Code:

```
import secrets
api_key = secrets.token_hex(16) # 32 characters
print(f"API Key: {api_key}") # e.g., "d3b07384d113edec49eaa6238ad5ff00"
```

❑ အဓိက ကွာခြားချက်များ

Feature	random Module	secrets Module
အသုံးပြုရန်	ယေဘုယျအသုံးပြုမှု	လုံခြုံရေးလိုအပ်သော အရာများ (passwords, tokens)
ထုတ်လုပ်မှု	သာမန် random	Cryptographically secure
Seed သတ်မှတ်နိုင်မှု	ရှိ	မရှိ

✍ Short Notes:

1. Passwords/Tokens အတွက် secrets module ကို အမြဲသုံးပါ။
2. random Module ကို စမ်းသပ်မှုများ သို့မဟုတ် လုံခြုံရေးမလိုအပ်သော အရာများအတွက်သာ သုံးပါ။
3. Reproducibility လိုအပ်ပါက random.seed() ကို သုံးပါ။



အခန်း(၁၂)

Regular Expression



❑ Regular Expression

Regular Expression (regex) ဆိုတာက string တွေကို search, match, replace လုပ်ဖို့အတွက် powerful tool တစ်ခုဖြစ်ပါတယ်။ Python မှာ regex ကို re module နဲ့ အသုံးပြုပါတယ်။ ဒီ tutorial မှာ regex အကြောင်းကို step by step ရှင်းပြပေးသွားမှာဖြစ်ပါတယ်။

၁၂.၁။ re Module ကို Import လုပ်ခြင်း

Python မှာ regex ကို အသုံးပြုဖို့အတွက် re module ကို import လုပ်ရပါမယ်။

Code:

```
import re
```

၁၂.၂။ Basic Regex Patterns

Regex မှာ အသုံးများတဲ့ patterns တွေကို အောက်မှာ ဖော်ပြထားပါတယ်။

- . : Any character (except newline)
- \d : Digit (0-9)
- \D : Not a digit
- \w : Word character (a-z, A-Z, 0-9, _)
- \W : Not a word character
- \s : Whitespace (space, tab, newline)
- \S : Not whitespace
- ^ : Start of a string
- \$: End of a string
- * : 0 or more repetitions
- + : 1 or more repetitions



- `?` : 0 or 1 repetition
- `{n}` : Exactly n repetitions
- `{n,}` : n or more repetitions
- `{n,m}` : Between n and m repetitions

၁၂.၃။ re.match() Function

re.match() က string ရဲ့စတင်ချင်းမှာ pattern နဲ့ match ဖြစ်မဖြစ်ကို စစ်ဆေးပါတယ်။

Code:

```
import re
```

```
pattern = r"hello"  
text = "hello world"
```

```
match = re.match(pattern, text)
```

```
if match:
```

```
    print("Match found:", match.group())
```

```
else:
```

```
    print("No match")
```

Output:

```
Match found: hello
```

၁၂.၄။ re.search() Function

re.search() က string ထဲမှာ pattern နဲ့ match ဖြစ်တဲ့အပိုင်းကို ရှာဖွေပေးပါတယ်။



Code:

```
import re

pattern = r"world"
text = "hello world"

match = re.search(pattern, text)

if match:
    print("Match found:", match.group())
else:
    print("No match")
```

Output:

```
Match found: world
```

၁၂.၅။ re.findall() Function

re.findall() က string ထဲမှာ pattern နဲ့ match ဖြစ်တဲ့အပိုင်းတွေကို list အဖြစ် return ပြန်ပေးပါတယ်။

Code:

```
import re

pattern = r"\d+"
text = "There are 3 apples and 5 oranges."
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Output:

```
Matches: ['3', '5']
```

၁၂.၆။ re.sub() Function

re.sub() က string ထဲမှာ pattern နဲ့ match ဖြစ်တဲ့အပိုင်းကို replace လုပ်ပေးပါတယ်။

Code:

```
import re

pattern = r"apple"
text = "I have an apple."
new_text = re.sub(pattern, "banana", text)
print("New text:", new_text)
```

Output:

```
New text: I have an banana.
```

၁၂.၇။ Grouping and Capturing

Parentheses () ကို အသုံးပြုပြီး pattern တွေကို group လုပ်နိုင်ပါတယ်။

Code:

```
import re

pattern = r"(\d{2})-(\d{2})-(\d{4})"
text = "Date: 12-31-2022"

match = re.search(pattern, text)

if match:
    print("Full match:", match.group(0))
    print("Day:", match.group(1))
    print("Month:", match.group(2))
    print("Year:", match.group(3))
```

Output:

```
Full match: 12-31-2022
Day: 12
Month: 31
Year: 2022
```

၁၂.၈။ Using | (OR Operator)

| (or operator) ကို အသုံးပြုပြီး multiple patterns တွေကို match လုပ်နိုင်ပါတယ်။

Code:

```
import re

pattern = r"cat|dog"
text = "I have a cat and a dog."
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Output:

```
Matches: ['cat', 'dog']
```

၁၂.၉။ Using ^ and \$

^ က string ရဲ့စတင်နေရာမှာ match ဖြစ်မဖြစ်ကို စစ်ဆေးပြီး \$ က string ရဲ့နောက်ဆုံးမှာ match ဖြစ်မဖြစ်ကို စစ်ဆေးပါတယ်။

Code:

```
import re

pattern = r"^hello"
text = "hello world"
match = re.match(pattern, text)

if match:
    print("Match found:", match.group())
else:
    print("No match")
```

Output:

Match found: hello

၁၂.၁၀။ Using \b (Word Boundary)

\b က word boundary ကို ရည်ညွှန်းပါတယ်။

Code:

```
import re

pattern = r"\bworld\b"
text = "hello world"
match = re.search(pattern, text)

if match:
    print("Match found:", match.group())
else:
    print("No match")
```

Output:

Match found: world

၁၂.၁၁။ Using \d, \w, \s, etc.

\d က digit တွေကို match လုပ်ပြီး \w က word characters တွေကို match လုပ်ပါတယ်။

Code:

```
import re

pattern = r"\d+"
text = "There are 3 apples and 5 oranges."
matches = re.findall(pattern, text)

print("Matches:", matches)
```

Output:

```
Matches: ['3', '5']
```

၁၂.၁၂။ Using *, +, ?, {}

- *: 0 or more repetitions
- + : 1 or more repetitions
- ? : 0 or 1 repetition
- {n} : Exactly n repetitions

Code:

```
import re

pattern = r"a{2,4}"
text = "aa, aaa, aaaa"
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Output:

```
Matches: ['aa', 'aaa', 'aaaa']
```

၁၂.၁၃။ Using [] (Character Classes)

[] ကို အသုံးပြုပြီး specific characters တွေကို match လုပ်နိုင်ပါတယ်။

Code:

```
import re

pattern = r"[aeiou]"
text = "hello world"
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Output:

```
Matches: ['e', 'o', 'o']
```

၁၂.၁၄။ Using \ (Escape Character)

Special characters တွေကို match လုပ်ဖို့အတွက် \ ကို အသုံးပြုပါတယ်။

Code:

```
import re

pattern = r"\.w+"
text = "example.com"
match = re.search(pattern, text)
```

```
if match:
    print("Match found:", match.group())
else:
    print("No match")
```

Output:

Match found: .com

၁၂.၁၅။ Using re.IGNORECASE Flag

re.IGNORECASE flag ကို အသုံးပြုပြီး case-insensitive match လုပ်နိုင်ပါတယ်။

Code:

```
import re

pattern = r"hello"
text = "HELLO world"
match = re.search(pattern, text, re.IGNORECASE)

if match:
    print("Match found:", match.group())
else:
    print("No match")
```

Output:

Match found: HELLO

၁၂.၁၆။ Using re.MULTILINE Flag

re.MULTILINE flag ကို အသုံးပြုပြီး multiline string တွေမှာ ^ နဲ့ \$ ကို line by line စစ်ဆေးနိုင်ပါတယ်။

Code:

```
import re
```

```
pattern = r"^hello"  
text = """hello world  
hello there  
hi hello"""
```

```
matches = re.findall(pattern, text, re.MULTILINE)  
print("Matches:", matches)
```

Output:

```
Matches: ['hello', 'hello']
```

၁၂.၁၇။ Using re.DOTALL Flag

re.DOTALL flag ကို အသုံးပြုပြီး . က newline (\n) ကိုပါ match လုပ်နိုင်ပါတယ်။

Code:

```
import re
```

```
pattern = r"hello.*world"  
text = """hello  
world"""  
match = re.search(pattern, text, re.DOTALL)
```

if match:

```
    print("Match found:", match.group())
```



```
else:
    print("No match")
```

Output:

```
Match found: hello
world
```

၁၂.၁၈။ Using re.VERBOSE Flag

re.VERBOSE flag ကို အသုံးပြုပြီး regex pattern ကို ပိုမိုဖတ်ရလွယ်ကူအောင် ရေးနိုင်ပါတယ်။

Code:

```
import re

pattern = r"""
\d{2} # Day
-    # Separator
\d{2} # Month
-    # Separator
\d{4} # Year
"""

text = "Date: 12-31-2022"
match = re.search(pattern, text, re.VERBOSE)

if match:
    print("Match found:", match.group())
else:
    print("No match")
```

Output:

```
Match found: 12-31-2022
```

၁၂.၁၉။ Using re.compile()

re.compile() ကို အသုံးပြုပြီး regex pattern ကို compile လုပ်နိုင်ပါတယ်။

Code:

```
import re

pattern = re.compile(r"\d+")
text = "There are 3 apples and 5 oranges."
matches = pattern.findall(text)
print("Matches:", matches)
```

Output:

```
Matches: ['3', '5']
```

၁၂.၂၀။ Using re.split()

re.split() ကို အသုံးပြုပြီး string ကို regex pattern အရ split လုပ်နိုင်ပါတယ်။

Code:

```
import re

pattern = r"\s+"
text = "Split this text by spaces."
result = re.split(pattern, text)
print("Split result:", result)
```

Output:

```
Split result: ['Split', 'this', 'text', 'by', 'spaces.']
```

Short Notes:

Python မှာ regular expressions တွေကို အသုံးပြုပြီး string တွေကို search, match, replace လုပ်ဖို့အတွက် အလွန်အသုံးဝင်ပါတယ်။ re module ကို အသုံးပြုပြီး မတူညီတဲ့ regex patterns တွေကို လေ့လာပြီး လိုအပ်တဲ့အခါမှာ အသုံးပြုနိုင်ပါတယ်။



လေ့ကျင့်ရန်-

Practice 1:

```
import re
```

```
str="hello this is my phone no 09449008972 and my sister's no is 09425029450 and  
my uncle no is 09425028458"
```

```
phone_list=re.findall('[\d]{11}',str)
```

```
for phone in phone_list:  
    print(phone)
```



Output:

```
09449008972
09425029450
09425028458
```

Practice 2:

```
import re

str="hello this is my email northerncity@gmail.com and my sister's email is
susu@gmail.com and my uncle no is aung77@gmail.com"
email_list=re.findall('[0-9a-zA-Z\.-]+@[a-zA-Z0-9\.-]+',str)

for email in email_list:
    print(email)
```

Output:

```
northerncity@gmail.com
susu@gmail.com
aung77@gmail.com
```

အခန်း(၁၃)

Object Oriented Programming



❑ OOP

OOP (Object-Oriented Programming) ကို အစအဆုံး အသေးစိတ်ရှင်းပြပေးပါမယ်။ OOP ဆိုတာက programming paradigm တစ်ခုဖြစ်ပြီး၊ real-world entities တွေကို objects အဖြစ် model လုပ်ပြီး program ရေးသားတဲ့နည်းလမ်းဖြစ်ပါတယ်။ Python မှာ OOP ကို class နဲ့ object တွေကိုအသုံးပြုပြီး implement လုပ်ပါတယ်။

၁၃.၁။ Class နဲ့ Object ကိုနားလည်ခြင်း

Class ဆိုတာက object တွေကို ဖန်တီးဖို့အတွက် blueprint တစ်ခုဖြစ်ပါတယ်။ Class ထဲမှာ attributes (data) နဲ့ methods (functions) တွေပါဝင်ပါတယ်။

Object ဆိုတာက class ကနေ ဖန်တီးထားတဲ့ instance တစ်ခုဖြစ်ပါတယ်။ Object တစ်ခုမှာ class ထဲက attributes နဲ့ methods တွေပါဝင်ပါတယ်။

၁၃.၂။ Class ဖန်တီးခြင်း

Python မှာ class ဖန်တီးဖို့ class keyword ကိုသုံးပါတယ်။

Code:

```
class Car:  
    pass
```

ဒီနမူနာမှာ Car ဆိုတဲ့ class တစ်ခုကိုဖန်တီးထားပါတယ်။ ဒါပေမယ့် ဒီ class ထဲမှာ attributes နဲ့ methods တွေမပါဝင်သေးပါဘူး။

၁၃.၃။ Attributes နဲ့ Methods ထည့်ခြင်း

Class ထဲမှာ attributes နဲ့ methods တွေထည့်လို့ရပါတယ်။

Code:

```
class Car:
    # Class attribute
    wheels = 4

    # Instance attributes
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    # Method
    def display_info(self):
        print(f"This car is a {self.brand} {self.model} with {self.wheels} wheels.")
```

ရှင်းလင်းချက်။

- wheels ဆိုတာက class attribute ဖြစ်ပါတယ်။ ဒါက class အတွင်းမှာ shared ဖြစ်ပါတယ်။
- __init__ method က constructor method ဖြစ်ပါတယ်။ Object ဖန်တီးတဲ့အခါမှာ automatically ခေါ်ပါတယ်။
- self ဆိုတာက instance ကိုရည်ညွှန်းပါတယ်။ self ကို method တွေထဲမှာ instance attributes တွေကိုရည်ညွှန်းဖို့သုံးပါတယ်။
- display_info method က instance method ဖြစ်ပါတယ်။

၁၃.၄။ Object ဖန်တီးခြင်း

Class ကနေ object တွေကိုဖန်တီးလို့ရပါတယ်။

Code:

```
# Creating objects
```

```
car1 = Car("Toyota", "Corolla")
```

```
car2 = Car("Honda", "Civic")
```

```
# Accessing attributes
```

```
print(car1.brand) # Output: Toyota
```

```
print(car2.model) # Output: Civic
```

```
# Calling methods
```

```
car1.display_info() # Output: This car is a Toyota Corolla with 4 wheels.
```

```
car2.display_info() # Output: This car is a Honda Civic with 4 wheels.
```

၁၃.၅။ Inheritance (အမွေဆက်ခံခြင်း)

Inheritance ဆိုတာက class တစ်ခုကနေ အခြား class တစ်ခုကို attributes နဲ့ methods တွေကို inherit လုပ်တာဖြစ်ပါတယ်။

Code:

```
class ElectricCar(Car):
```

```
    def __init__(self, brand, model, battery_capacity):
```

```
        super().__init__(brand, model)
```

```
        self.battery_capacity = battery_capacity
```

```
    def display_info(self):
```

```
        print(f"This electric car is a {self.brand} {self.model} with {self.wheels} wheels and a {self.battery_capacity} kWh battery.")
```

- ElectricCar class က Car class ကနေ inherit လုပ်ထားပါတယ်။
- super().__init__(brand, model) က parent class ရဲ့ __init__ method ကိုခေါ်ပါတယ်။
- display_info method ကို override လုပ်ထားပါတယ်။

Code:

```
# Creating an object of ElectricCar
electric_car = ElectricCar("Tesla", "Model S", 100)

# Calling methods
electric_car.display_info() # Output: This electric car is a Tesla Model S with 4 wheels and a 100 k
Wh battery.
```

၁၃.၆။ Encapsulation (ဖုံးကွယ်ခြင်း)

Encapsulation ဆိုတာက data နဲ့ methods တွေကို class ထဲမှာ ဖုံးကွယ်ထားတာဖြစ်ပါတယ်။ Python မှာ encapsulation ကို private attributes တွေနဲ့ implement လုပ်ပါတယ်။

Code:

```
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self.__balance = balance # Private attribute

    def deposit(self, amount):
        self.__balance += amount

    def withdraw(self, amount):
        if amount > self.__balance:
            print("Insufficient funds")
        else:
            self.__balance -= amount

    def get_balance(self):
        return self.__balance
```

ရှင်းလင်းချက်။

- `__balance` ဆိုတာက private attribute ဖြစ်ပါတယ်။ Class ပြင်ပကနေ တိုက်ရိုက်မရနိုင်ပါဘူး။
- `get_balance` method ကို အသုံးပြုပြီး `balance` ကိုရယူနိုင်ပါတယ်။

Code:

```
# Creating an object
account = BankAccount("John", 1000)

# Accessing private attribute (will raise an error)
# print(account.__balance) # AttributeError

# Using methods to interact with private attribute
account.deposit(500)
print(account.get_balance()) # Output: 1500

account.withdraw(200)
print(account.get_balance()) # Output: 1300
```

၁၃.၇။ Polymorphism (ပုံစံအမျိုးမျိုးဖြစ်ခြင်း)

Polymorphism ဆိုတာက same interface ကို အသုံးပြုပြီး different implementations တွေကို လုပ်ဆောင်နိုင်တာဖြစ်ပါတယ်။

Code:

```
class Dog:
    def sound(self):
        return "Woof!"

class Cat:
    def sound(self):
        return "Meow!"

# Polymorphism in action
def make_sound(animal):
    print(animal.sound())
```

```
dog = Dog()
cat = Cat()
```

```
make_sound(dog) # Output: Woof!
make_sound(cat) # Output: Meow!
```

ရှင်းလင်းချက်။

- Dog နဲ့ Cat class တွေမှာ sound method တွေရှိပါတယ်။
- make_sound function ကို သုံးပြီး different objects တွေရဲ့ sound method တွေကို call လုပ်နိုင်ပါတယ်။

၁၃.၈။ Abstraction

Abstraction ဆိုတာက complex implementation details တွေကို ဖုံးကွယ်ပြီး၊ simple interface တစ်ခုကိုပဲပြသထားတာဖြစ်ပါတယ်။ Python မှာ abstraction ကို abstract base classes တွေနဲ့ implement လုပ်နိုင်ပါတယ်။

Code:

```
from abc import ABC, abstractmethod
```

```
class Animal(ABC):
    @abstractmethod
    def sound(self):
        pass
```

```
class Dog(Animal):
    def sound(self):
        return "Woof!"
```

```
class Cat(Animal):
    def sound(self):
        return "Meow!"
```

```
# Creating objects
dog = Dog()
```

```
cat = Cat()
```

```
print(dog.sound()) # Output: Woof!
```

```
print(cat.sound()) # Output: Meow!
```

ရှင်းလင်းချက်။

- Animal class က abstract base class ဖြစ်ပါတယ်။
- sound method ကို @abstractmethod decorator နဲ့ define လုပ်ထားပါတယ်။
- Dog နဲ့ Cat class တွေက Animal class ကနေ inherit လုပ်ပြီး sound method ကို implement လုပ်ထားပါတယ်။

Short Notes:

Python OOP ကို အသုံးပြုပြီး real-world entities တွေကို model လုပ်နိုင်ပါတယ်။ Class, Object, Inheritance, Encapsulation, Polymorphism, နဲ့ Abstraction တွေကို နားလည်ပြီး သုံးတတ်ရင် complex programs တွေကို ပိုမိုကောင်းမွန်စွာ ရေးသားနိုင်မှာဖြစ်ပါတယ်။

အခန်း(၁၄)

Interface



❑ Interface

Interface ဆိုတာ class တွေအတွက် သတ်မှတ်ချက် (contract) တစ်ခုဖြစ်ပြီး၊ ဘယ် method တွေ ပါရမယ်ဆိုတာကို သတ်မှတ်ပေးပါတယ်။ Python မှာ Interface ကို **Abstract Base Classes (ABCs)** နဲ့ တည်ဆောက်လေ့ရှိပါတယ်။

၁၄.၁။ Interface ၏ အခြေခံသဘောတရား

- Interface သည် method signatures (method အမည်နှင့် parameters) သာသတ်မှတ်ပြီး၊ လက်တွေ့ implementation မပါပါ။
 - Interface ကို implement လုပ်သော class များက သတ်မှတ်ထားသော method များအားလုံးကို လက်တွေ့အသုံးပြုရပါမယ်။
-

၁၄.၂။ Python မှာ Interface ကိုဘယ်လိုတည်ဆောက်မလဲ

Python မှာ Interface ကို abc module မှ ABC class နှင့် @abstractmethod decorator ကိုသုံးပြီးတည်ဆောက်ပါတယ်။

Example: ရိုးရှင်းသော Interface

Code:

```
from abc import ABC, abstractmethod
```

```
# Interface ကိုသတ်မှတ်ခြင်း
```

```
class Shape(ABC):
```

```
    @abstractmethod
```

```
def area(self):
    pass

@abstractmethod
def perimeter(self):
    pass

# Interface ကို implement လုပ်သော Class
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2

    def perimeter(self):
        return 2 * 3.14 * self.radius

# Interface ကို implement လုပ်သော Class
class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)

# Usage
circle = Circle(5)
print("Circle Area:", circle.area()) # Output: 78.5
rectangle = Rectangle(4, 5)
print("Rectangle Perimeter:", rectangle.perimeter()) # Output: 18
```


❑ အရေးကြီးသောအချက်များ:

1. Shape class သည် **Interface** ဖြစ်ပြီး ABC မှအမွေဆက်ခံထားသည်။
2. @abstractmethod decorator ဖြင့် method များကို abstract အဖြစ်သတ်မှတ်သည်။
3. Circle နှင့် Rectangle class များက Shape interface ကို implement လုပ်ပြီး abstract method များအားလုံးကို override လုပ်ရပါမည်။

၁၄.၃။ Interface ကို မလိုက်နာပါက Error ဖြစ်ခြင်း

Interface မှသတ်မှတ်ထားသော method တစ်ခုကို implement မလုပ်ပါက Python က TypeError ကိုပြသပါမည်။

Code:

```
class Triangle(Shape):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    # area() method ကို implement မလုပ်ထား
    def perimeter(self):
        return self.a + self.b + self.c

# Error ဖြစ်မယ်
# triangle = Triangle(3,4,5) # TypeError: Can't instantiate abstract class Triangle with abstract method area
```

၁၄.၄။ Multiple Interfaces (Interface များ)

Python မှာ class တစ်ခုသည် **Multiple Interfaces** များကို implement လုပ်နိုင်ပါတယ်။

Example: နှစ်ခုသော Interfaces များ

Code:

```
from abc import ABC, abstractmethod

# Interface 1
class Drawable(ABC):
    @abstractmethod
    def draw(self):
        pass

# Interface 2
class Clickable(ABC):
    @abstractmethod
    def click(self):
        pass

# Class က Interfaces နှစ်ခုလုံးကို implement လုပ်ခြင်း
class Button(Drawable, Clickable):
    def draw(self):
        print("Button is drawn")

    def click(self):
        print("Button is clicked")

# Usage
button = Button()
button.draw() # Output: Button is drawn
button.click() # Output: Button is clicked
```

၁၄.၅။ Interface vs Abstract Class

- Interface: Method signatures သာပါပြီး implementation မပါ။
- Abstract Class: Method implementations နှင့် attributes များပါနိုင်ပါတယ်။

Example: Abstract Class နှင့် Interface ကွာခြားချက်

Code:

```
from abc import ABC, abstractmethod

# Abstract Class
class DatabaseConnector(ABC):
    @abstractmethod
    def connect(self):
        pass

    def close(self):
        print("Closing database connection")

# Interface
class Queryable(ABC):
    @abstractmethod
    def execute_query(self, query):
        pass

# Class က Abstract Class နှင့် Interface ကို implement လုပ်ခြင်း
class MySQLConnector(DatabaseConnector, Queryable):
    def connect(self):
        print("Connected to MySQL")

    def execute_query(self, query):
        print(f"Executing query: {query}")

# Usage
```



```
mysql = MySQLConnector()
mysql.connect()      # Output: Connected to MySQL
mysql.execute_query("SELECT * FROM users") # Output: Executing query: SELECT * FROM users
mysql.close()        # Output: Closing database connection
```

✍ Short Notes:

- Python မှာ Interface ကို abc module နှင့် @abstractmethod တို့ဖြင့်တည်ဆောက်ပါ။
- Interface သည် class များအား method များကိုအတင်းအကျပ်လိုက်နာစေပါတယ်။
- Duck Typing နှင့် Interface တို့ကို လိုအပ်သလိုပေါင်းစပ်အသုံးပြုနိုင်ပါတယ်။

အခန်း(၁၅)

File Handling





❑ File Handling

Python မှာ File Handling ဆိုတာက program ထဲကနေ files တွေကို read/write လုပ်တဲ့ process ဖြစ်ပါတယ်။ ဒီ guide မှာ file operations တွေကို အဆင့်ဆင့်ရှင်းပြပြီး example code တွေနဲ့ပြသပေးမှာဖြစ်ပါတယ်။

၁၅.၁။ File ဖွင့်ခြင်း (Opening Files)

Python မှာ file တစ်ခုကို အသုံးပြုဖို့အတွက် `open()` function ကိုသုံးပါတယ်။ ဖိုင်အမျိုးအစားနဲ့ mode တွေကို သတ်မှတ်နိုင်ပါတယ်။

❑ Modes အမျိုးမျိုး:

- r - Read mode (default)
- w - Write mode (ဖိုင်ရှိရင် overwrite၊ မရှိရင် အသစ်ဖန်တီး)
- a - Append mode (ဖိုင်ရှိရင် အဆုံးမှာ ထပ်ထည့်)
- r+ - Read + Write mode
- b - Binary mode (e.g., rb, wb)

Syntax:

```
file_object = open("filename.txt", "mode")
```

Example 1: File ဖွင့်ပြီး စာသားရေးခြင်း

Code:

```
# Write mode နဲ့ဖွင့်ပြီး စာသားရေးမယ်
file = open("demo.txt", "w")
file.write("Hello Python File Handling!\n")

file.close() # ဖိုင်ကို ပိတ်ရန်
```

၁၅.၂။ File ထဲမှ စာသားဖတ်ခြင်း (Reading Files)

ဖိုင်ထဲကစာသားတွေကို ဖတ်ဖို့ read(), readline(), readlines() method တွေကိုသုံးပါတယ်။

Example: ဖိုင်တစ်ခုလုံးကို ဖတ်ခြင်း

Code:

```
file = open("demo.txt", "r")
content = file.read()
print(content)
file.close()
```

Output: Hello Python File Handling!

Example: စာကြောင်းတစ်ကြောင်းချင်းဖတ်ခြင်း

Code:

```
file = open("demo.txt", "r")
print(file.readline()) # Output: Hello Python File Handling!
file.close()
```

Example: စာကြောင်းအားလုံးကို list အဖြစ်ဖတ်ခြင်း

Code:

```
file = open("demo.txt", "r")
lines = file.readlines()
print(lines) # Output: ['Hello Python File Handling!\n']
file.close()
```

၁၅.၃။ File ကို Append လုပ်ခြင်း (Appending Data)

a mode ကိုသုံးပြီး ဖိုင်ရဲ့အဆုံးမှာ စာသားထပ်ထည့်နိုင်ပါတယ်။

Example: ဖိုင်ကို ထပ်ဖြည့်ခြင်း

Code:

```
file = open("demo.txt", "a")
file.write("This is a new line.\n")
file.close()
```

ဖိုင်ထဲကစာသားအသစ်:

Output:

Hello Python File Handling!
This is a new line.

၁၅.၄။ with Statement ကိုသုံးခြင်း

with statement ကို သုံးရင် file ကို auto-close လုပ်ပေးပါတယ်။ ပိုစိတ်ချရပါတယ်။

Example: with statement နဲ့ ဖိုင်ဖတ်ခြင်း

Code:

```
with open("demo.txt", "r") as file:  
    content = file.read()  
    print(content)
```

၁၅.၅။ Binary Files ကိုအသုံးပြုခြင်း

Binary files (ဥပမာ images, videos) တွေကို b mode နဲ့အသုံးပြုပါတယ်။

Example: Binary file ကို copy ကူးခြင်း

Code:

```
with open("source.jpg", "rb") as source_file:  
    with open("copy.jpg", "wb") as dest_file:  
        dest_file.write(source_file.read())
```

၁၅.၆။ File Handling Best Practices

⇒ ၁၅.၆.၁။ ဖိုင်ရှိမရှိ စစ်ခြင်း

os.path module ကိုသုံးပြီး ဖိုင်ရှိမရှိ စစ်ပါ။

Code:

```
import os  
  
if os.path.exists("demo.txt"):  
    print("File exists!")  
else:  
    print("File not found!")
```

၁၅.၁.၂။ Exception Handling

try-except block နဲ့ errors တွေကို handle လုပ်ပါ။

Code:

try:

```
with open("nonexistent.txt", "r") as file:
```

```
    print(file.read())
```

except FileNotFoundError:

```
    print("File does not exist!")
```

except Exception as e:

```
    print(f"An error occurred: {e}")
```

၁၅.၂။ CSV နှင့် JSON Files များကို အသုံးပြုခြင်း

Python မှာ built-in modules တွေဖြစ်တဲ့ csv နဲ့ json တို့ကို အသုံးပြုပြီး structured data တွေကို handle လုပ်နိုင်ပါတယ်။

Example: CSV File ဖတ်ခြင်း

Code:

```
import csv
```

```
with open("data.csv", "r") as file:
```

```
    reader = csv.reader(file)
```

```
    for row in reader:
```

```
        print(row)
```

Example: JSON File ရေးခြင်း

Code:

```
import json
```

```
data = {"name": "Alice", "age": 30}
```

```
with open("data.json", "w") as file:
```

```
    json.dump(data, file)
```



✍ Short Notes:

1. File Modes - r, w, a စတဲ့ modes တွေကို လိုအပ်သလိုသုံးပါ။
2. with Statement - Auto-closing အတွက် သုံးပါ။
3. Exception Handling - မှီခိုတဲ့ error တွေကို handle လုပ်ပါ။
4. Libraries - CSV/JSON မှီခိုတဲ့အတွက် built-in modules တွေကို အသုံးပြုပါ။

Python file handling ကို ဒီနည်းနဲ့ အဆင်ပြေပြေအသုံးပြုနိုင်ပါတယ်။ လက်တွေ့ project တွေမှာ ဒီ concept တွေကို ပုံစံအမျိုးမျိုးနဲ့ တွေ့ရမှာဖြစ်ပါတယ်။



လေ့ကျင့်ရန်-

❑ Yoma Bank ATM Console Application

Project Structure

yoma_bank_atm/

- ├── main.py - Main application logic
- ├── accounts.dat - User account data file
- └── transactions.dat - Transaction history file

❑ Main Application Code (main.py)

```
import os
import json
from datetime import datetime

# File paths
ACCOUNTS_FILE = "accounts.dat"
TRANSACTIONS_FILE = "transactions.dat"

def load_data():
    """Load account and transaction data from files"""
    accounts = {}
    transactions = []

    try:
        with open(ACCOUNTS_FILE, 'r') as f:
            accounts = json.load(f)
    except (FileNotFoundError, json.JSONDecodeError):
        accounts = {}

    try:
        with open(TRANSACTIONS_FILE, 'r') as f:
            transactions = json.load(f)
    except (FileNotFoundError, json.JSONDecodeError):
        transactions = []

    return accounts, transactions

def save_data(accounts, transactions):
    """Save data to files"""
    with open(ACCOUNTS_FILE, 'w') as f:
        json.dump(accounts, f)

    with open(TRANSACTIONS_FILE, 'w') as f:
        json.dump(transactions, f)

def create_account(accounts):
    """Create new bank account"""
```

```
print("\n----- အကောင့်အသစ်ဖွင့်ရန် -----")
name = input("ကျေးဇူးပြု၍ သင့်အမည်ထည့်ပါ: ")
pin = input("PIN နံပါတ် (ဂဏန်း ၄ လုံး) ထည့်ပါ: ")

if len(pin) != 4 or not pin.isdigit():
    print("PIN နံပါတ်သည် ဂဏန်း ၄ လုံးဖြစ်ရပါမယ်")
    return

account_no = str(len(accounts) + 100000)
accounts[account_no] = {
    'name': name,
    'pin': pin,
    'balance': 0
}

print(f"\nကျေးဇူးတင်ပါသည် {name} သင်၏ အကောင့်နံပါတ်မှာ {account_no} ဖြစ်ပါတယ်")
return account_no

def login(accounts):
    """User login function"""
    print("\n----- အကောင့်ဝင်ရန် -----")
    account_no = input("အကောင့်နံပါတ်ထည့်ပါ: ")
    pin = input("PIN နံပါတ်ထည့်ပါ: ")

    if account_no in accounts and accounts[account_no]['pin'] == pin:
        print(f"\nမင်္ဂလာပါ {accounts[account_no]['name']} သင်")
        return account_no
    else:
        print("\nအကောင့်နံပါတ် (သို့) PIN မှားယွင်းနေပါတယ်")
        return None

def show_balance(account_no, accounts):
    """Show account balance"""
    balance = accounts[account_no]['balance']
    print(f"\nသင့်အကောင့်ရှိ လက်ကျန်ငွေမှာ {balance:,} ကျပ်ဖြစ်ပါတယ်")
```

```
def deposit(account_no, accounts, transactions):
    """Deposit money"""
    try:
        amount = float(input("\nထည့်ငွေပမာဏထည့်ပါ: "))
        if amount <= 0:
            print("ကျေးဇူးပြု၍ ၀ ထက်ကြီးသော ပမာဏထည့်ပါ")
            return

        accounts[account_no]['balance'] += amount
        transactions.append({
            'account': account_no,
            'type': 'deposit',
            'amount': amount,
            'date': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        })

        print(f"\n{amount:,} ကျပ်ထည့်ငွေအောင်မြင်ပါတယ်")
        show_balance(account_no, accounts)
    except ValueError:
        print("ကျေးဇူးပြု၍ ငွေပမာဏကို ဂဏန်းဖြင့်သာထည့်ပါ")

def withdraw(account_no, accounts, transactions):
    """Withdraw money"""
    try:
        amount = float(input("\nထုတ်ယူမည့်ပမာဏထည့်ပါ: "))
        balance = accounts[account_no]['balance']

        if amount <= 0:
            print("ကျေးဇူးပြု၍ ၀ ထက်ကြီးသော ပမာဏထည့်ပါ")
            return

        if amount > balance:
            print("သင့်အကောင့်တွင် လုံလောက်သောငွေပမာဏမရှိပါ")
            return

        accounts[account_no]['balance'] -= amount
        transactions.append({
            'account': account_no,
```

```
'type': 'withdraw',
'amount': amount,
'date': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
})

print(f"\n{amount:,} ကျပ်ငွေထုတ်ယူမှုအောင်မြင်ပါတယ်")
show_balance(account_no, accounts)
except ValueError:
    print("ကျေးဇူးပြု၍ ငွေပမာဏကို ဂဏန်းဖြင့်သာထည့်ပါ")

def show_transactions(account_no, transactions):
    """Show transaction history"""
    print("\n----- ငွေလွှဲမှုမှတ်တမ်း -----")
    user_transactions = [t for t in transactions if t['account'] == account_no]

    if not user_transactions:
        print("ငွေလွှဲမှုမှတ်တမ်းမရှိသေးပါ")
        return

    for t in user_transactions:
        print(f"{t['date']} - {t['type']}: {t['amount']:,} ကျပ်")

def main():
    """Main application function"""
    accounts, transactions = load_data()

    while True:
        print("\n----- Yoma Bank ATM -----")
        print("1. အကောင့်အသစ်ဖွင့်ရန်")
        print("2. အကောင့်ဝင်ရန်")
        print("3. ထွက်ရန်")

        choice = input("ရွေးချယ်မှုနံပါတ်ထည့်ပါ: ")

        if choice == "1":
            create_account(accounts)
            save_data(accounts, transactions)
        elif choice == "2":
```

```
account_no = login(accounts)
if account_no:
    while True:
        print("\n----- ATM Menu -----")
        print("1. လက်ကျန်ငွေကြည့်ရန်")
        print("2. ငွေထည့်ရန်")
        print("3. ငွေထုတ်ရန်")
        print("4. ငွေလွှဲမှုမှတ်တမ်း")
        print("5. အကောင့်မှထွက်ရန်")

        option = input("ရွေးချယ်မှုနံပါတ်ထည့်ပါ: ")

        if option == "1":
            show_balance(account_no, accounts)
        elif option == "2":
            deposit(account_no, accounts, transactions)
            save_data(accounts, transactions)
        elif option == "3":
            withdraw(account_no, accounts, transactions)
            save_data(accounts, transactions)
        elif option == "4":
            show_transactions(account_no, transactions)
        elif option == "5":
            break
        else:
            print("မှားယွင်းသောရွေးချယ်မှုဖြစ်ပါတယ်")

    elif choice == "3":
        print("\nYoma Bank ATM ကိုအသုံးပြုတဲ့အတွက်ကျေးဇူးတင်ပါတယ်")
        break
    else:
        print("မှားယွင်းသောရွေးချယ်မှုဖြစ်ပါတယ်")

if __name__ == "__main__":
    main()
```




Application Testing Guide

1. First Run (No Data Files Exist)

----- Yoma Bank ATM -----

1. အကောင့်အသစ်ဖွင့်ရန်

2. အကောင့်ဝင်ရန်

3. ထွက်ရန်

ရွေးချယ်မှုနံပါတ်ထည့်ပါ: 1

----- အကောင့်အသစ်ဖွင့်ရန် -----

ကျေးဇူးပြု၍ သင့်အမည်ထည့်ပါ: မောင်မောင်

PIN နံပါတ် (၈ဇာန်း ၄ လုံး) ထည့်ပါ: 1234

ကျေးဇူးတင်ပါသည် မောင်မောင် သင်၏ အကောင့်နံပါတ်မှာ 100000 ဖြစ်ပါတယ်

2. Login and Deposit

Copy

Download

----- Yoma Bank ATM -----

1. အကောင့်အသစ်ဖွင့်ရန်

2. အကောင့်ဝင်ရန်

3. ထွက်ရန်

ရွေးချယ်မှုနံပါတ်ထည့်ပါ: 2

----- အကောင့်ဝင်ရန် -----

အကောင့်နံပါတ်ထည့်ပါ: 100000

PIN နံပါတ်ထည့်ပါ: 1234

မင်္ဂလာပါ မောင်မောင် သင်



----- ATM Menu -----

1. လက်ကျန်ငွေကြည့်ရန်

2. ငွေထည့်ရန်

3. ငွေထုတ်ရန်

4. ငွေလွှဲမှုမှတ်တမ်း

5. အကောင့်မှထွက်ရန်

ရွေးချယ်မှုနံပါတ်ထည့်ပါ: 2

ထည့်ငွေပမာဏထည့်ပါ: 500000

500,000.0 ကျပ်ထည့်ငွေအောင်မြင်ပါတယ်

သင့်အကောင့်ရှိ လက်ကျန်ငွေမှာ 500,000.0 ကျပ်ဖြစ်ပါတယ်

3. Check Balance and Withdraw

----- ATM Menu -----

1. လက်ကျန်ငွေကြည့်ရန်

2. ငွေထည့်ရန်

3. ငွေထုတ်ရန်

4. ငွေလွှဲမှုမှတ်တမ်း

5. အကောင့်မှထွက်ရန်

ရွေးချယ်မှုနံပါတ်ထည့်ပါ: 1

သင့်အကောင့်ရှိ လက်ကျန်ငွေမှာ 500,000.0 ကျပ်ဖြစ်ပါတယ်

----- ATM Menu -----

1. လက်ကျန်ငွေကြည့်ရန်

2. ငွေထည့်ရန်

3. ငွေထုတ်ရန်

4. ငွေလွှဲမှုမှတ်တမ်း

5. အကောင့်မှထွက်ရန်

ရွေးချယ်မှုနံပါတ်ထည့်ပါ: 3



ထုတ်ယူမည့်ပမာဏထည့်ပါ: 200000

200,000.0 ကျပ်ငွေထုတ်ယူမှုအောင်မြင်ပါတယ်

သင့်အကောင့်ရှိ လက်ကျန်ငွေမှာ 300,000.0 ကျပ်ဖြစ်ပါတယ်

4. View Transaction History

----- ATM Menu -----

1. လက်ကျန်ငွေကြည့်ရန်
2. ငွေထည့်ရန်
3. ငွေထုတ်ရန်
4. ငွေလွှဲမှုမှတ်တမ်း
5. အကောင့်မှထွက်ရန်

ရွေးချယ်မှုနံပါတ်ထည့်ပါ: 4

----- ငွေလွှဲမှုမှတ်တမ်း -----

2023-11-15 14:30:45 - deposit: 500,000.0 ကျပ်

2023-11-15 14:31:22 - withdraw: 200,000.0 ကျပ်

ဒီ application ကို run ဖို့အတွက် Python 3.x လိုအပ်ပါတယ်။ command line မှာ python main.py လို့ရိုက်ထည့်ပြီး run နိုင်ပါတယ်။

အခန်း (၁၆)

Jupyter Notebook Installation





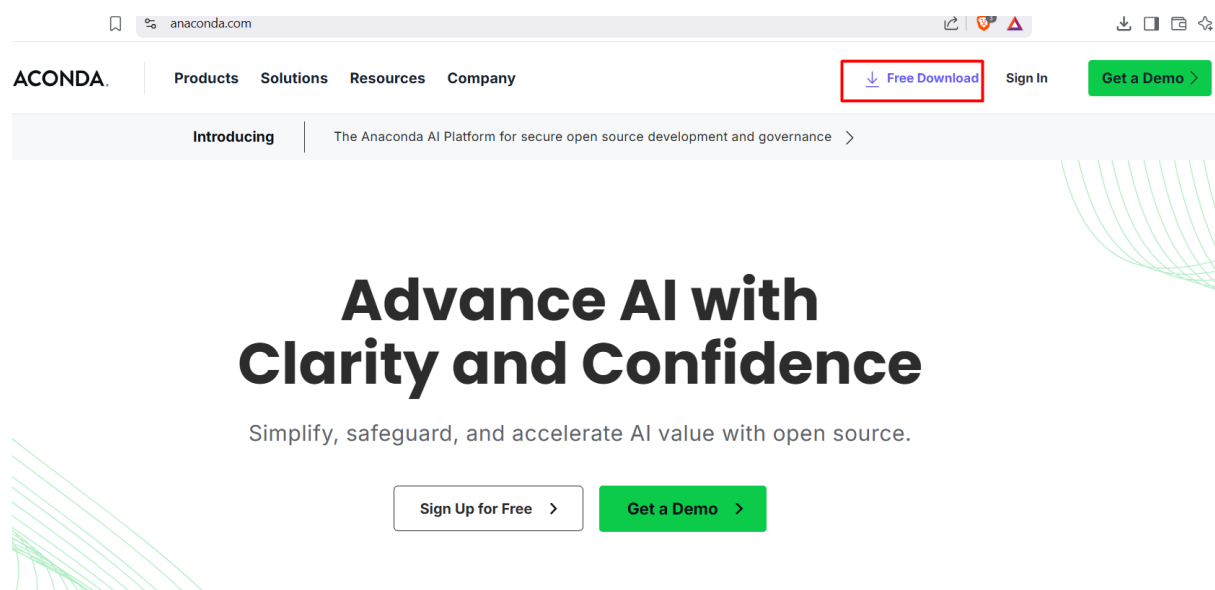
Jupyter Notebook ကိုသွင်းတော့မယ်ဆိုင်ရင် လိုအပ်တဲ့ System Requirement က အောက်ပါအတိုင်းဖြစ်ပါတယ်။

System Requirement:

1. Python 3.7 or later
2. Windows 7 or later
3. RAM: 4GB (atleast), 8GB or more is recommended
4. Storage: 1GB or more

Step 1:

www.anaconda.com website ကို သွားပြီး အောက်ပါအတိုင်း application ကို အဆင့်ဆင့် လုပ်ဆောင်ပြီး ဒေါင်းရပါမယ်။





Email ကို ဖြည့်လို့ရတယ်။ ကျော်သွားလို့လည်း ရပါတယ်။

anaconda.com/download

ANACONDA Products Solutions **Resources** Company [Free Download](#) [Sign In](#) [Get a Demo](#)

Distribution

FREE DOWNLOAD*

Register to get everything you need to get started on your workstation including Cloud Notebooks, Navigator, AI Assistant, Learning and more.

- ✓ Easily search and install thousands of data science, machine learning, and AI packages
- ✓ Manage packages and environments from a desktop application or work from the command line
- ✓ Deploy across hardware and software platforms
- ✓ Distribution installation on Windows, MacOS, or Linux

*Use of Anaconda's Offerings at an organization of more than 200 employees requires a Business or Enterprise license. [See Pricing](#)

Provide email to download Distribution

Email Address:
zautingmai@gmail.com

☒ Agree to receive communication from Anaconda regarding relevant content, products, and services. I understand that I can revoke this consent [here](#) at any time.

By continuing, I agree to Anaconda's [Privacy Policy](#) and [Terms of Service](#).

[Submit](#)

[Skip registration](#)

anaconda.com/download/success

ANACONDA Products Solutions Resources Company [Sign In](#) [Get a Demo](#)

Home

Download Now

Download Anaconda Distribution or Miniconda by choosing the proper installer for your machine. Learn the difference from our [Documentation](#).

Distribution Installers

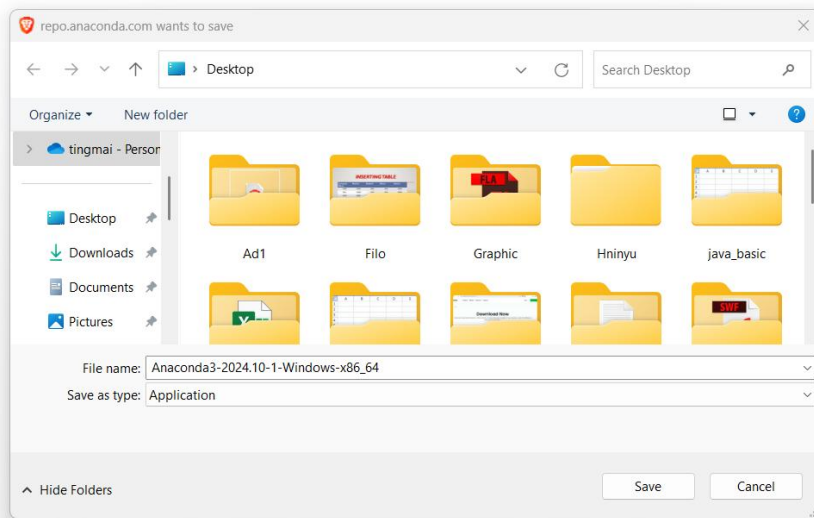
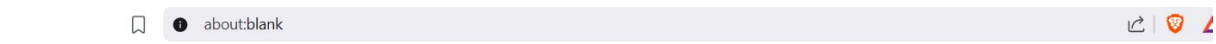
[Download](#)

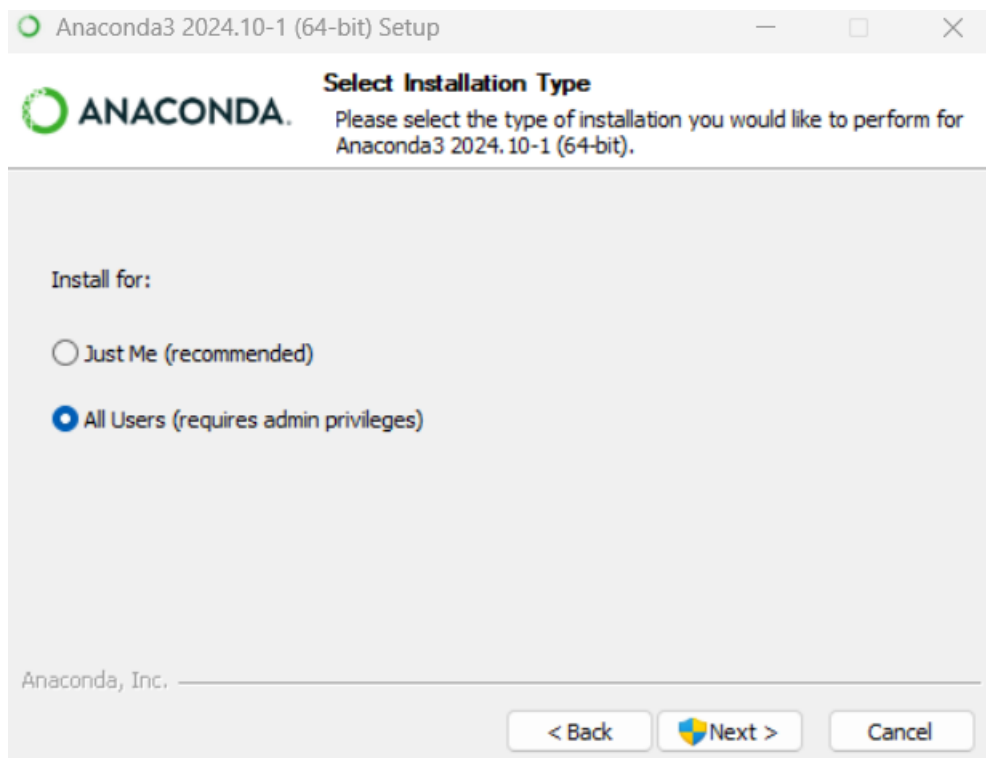
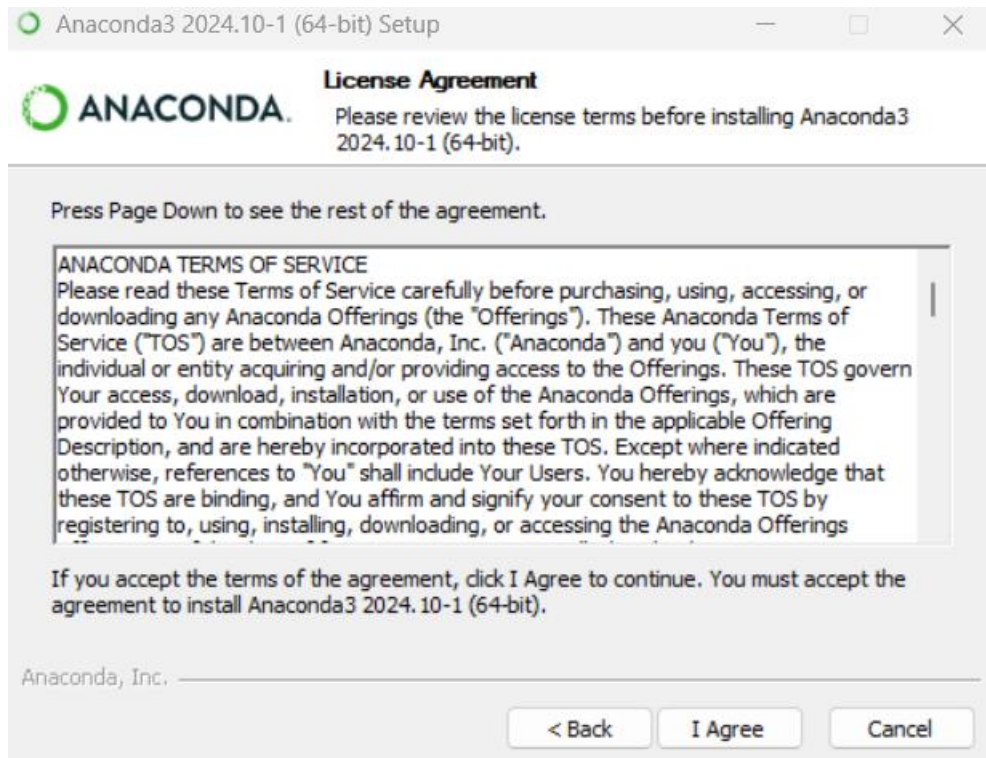
For installation assistance, refer to [troubleshooting](#).

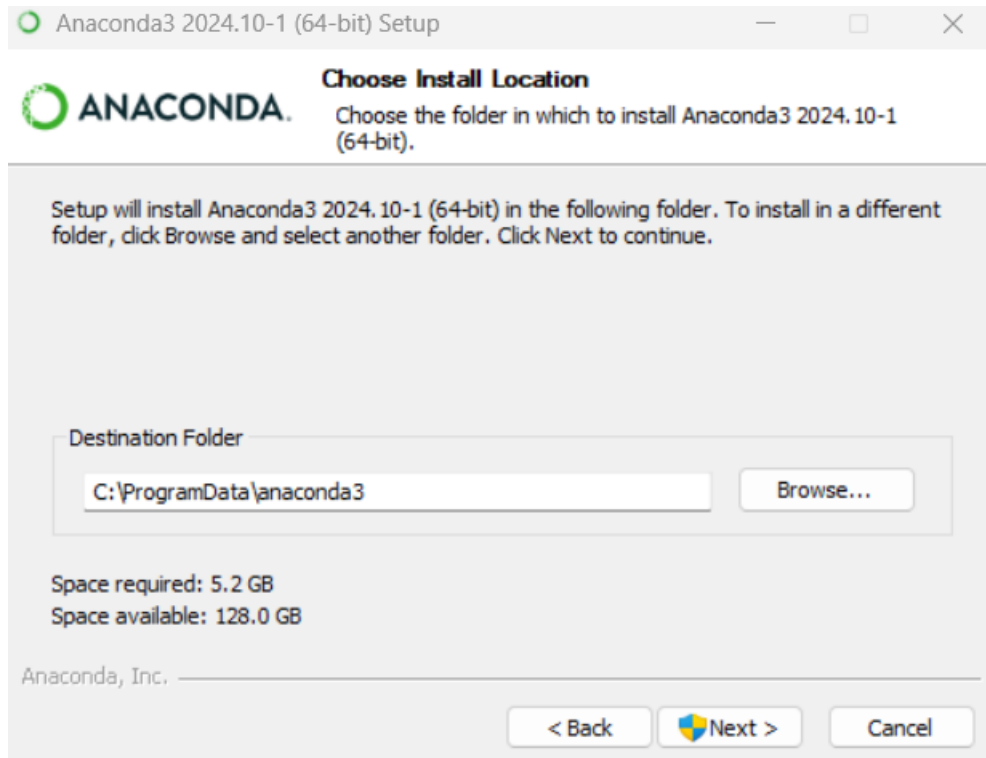
Miniconda Installers

[Download](#)

For installation assistance, refer to [troubleshooting](#).







Next Button တွေကို ဆက်တိုင်းနှိပ်သွားလိုက်ပါ။ installation ပြီးသွားရင် finished button ပေါ်လာပါလိမ့်မယ်။



Command prompt ထဲမှာ အောက်ပါအတိုင်း စစ်ဆေးနိုင်ပါတယ်။

```
Anaconda Prompt

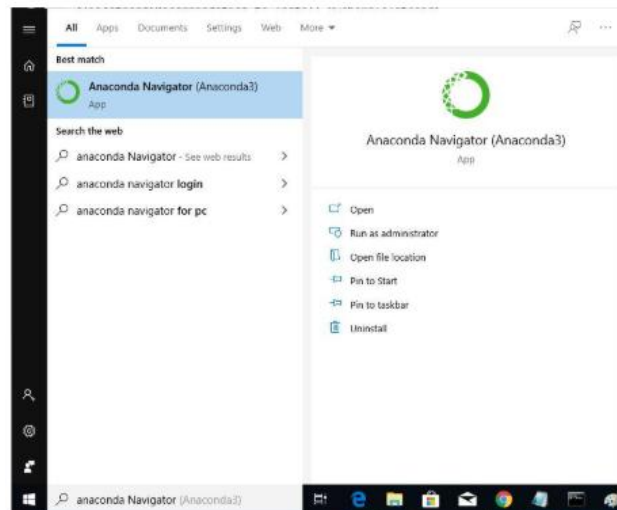
(base) C:\Users\SHARVA>conda --version
conda 23.3.1

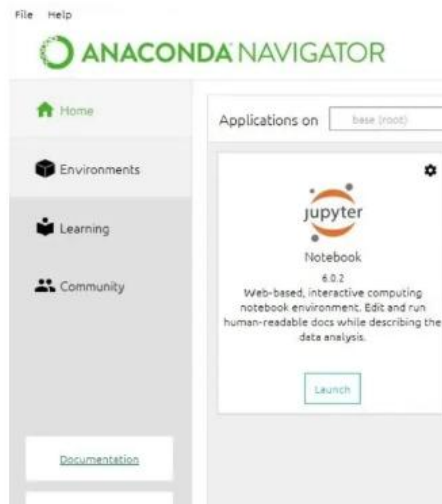
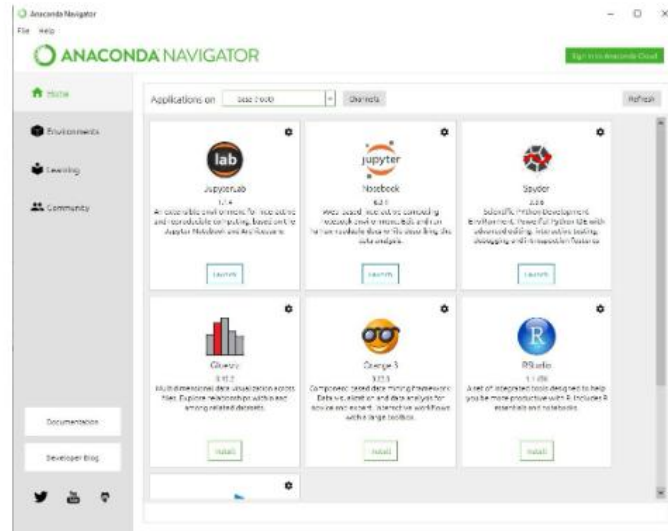
(base) C:\Users\SHARVA>python --version
Python 3.10.9

(base) C:\Users\SHARVA>_
```

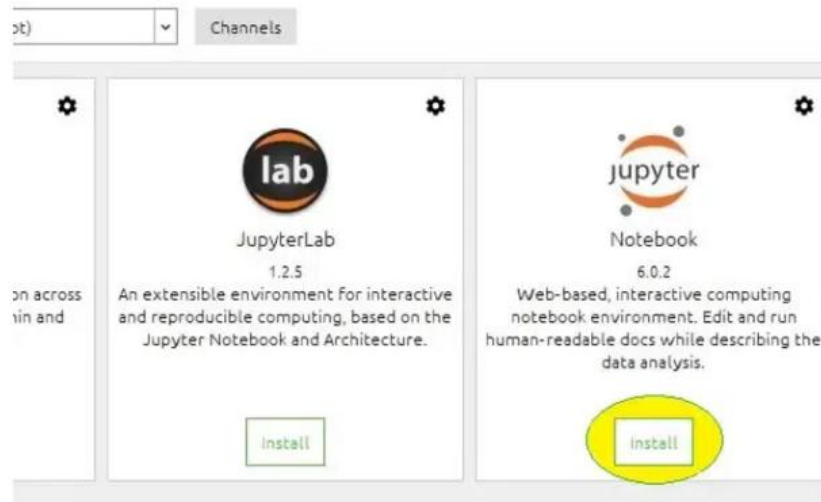
Verify the installer

Program ထဲက application ကို အောက်ပါအတိုင်း run ပေးရပါမယ်။

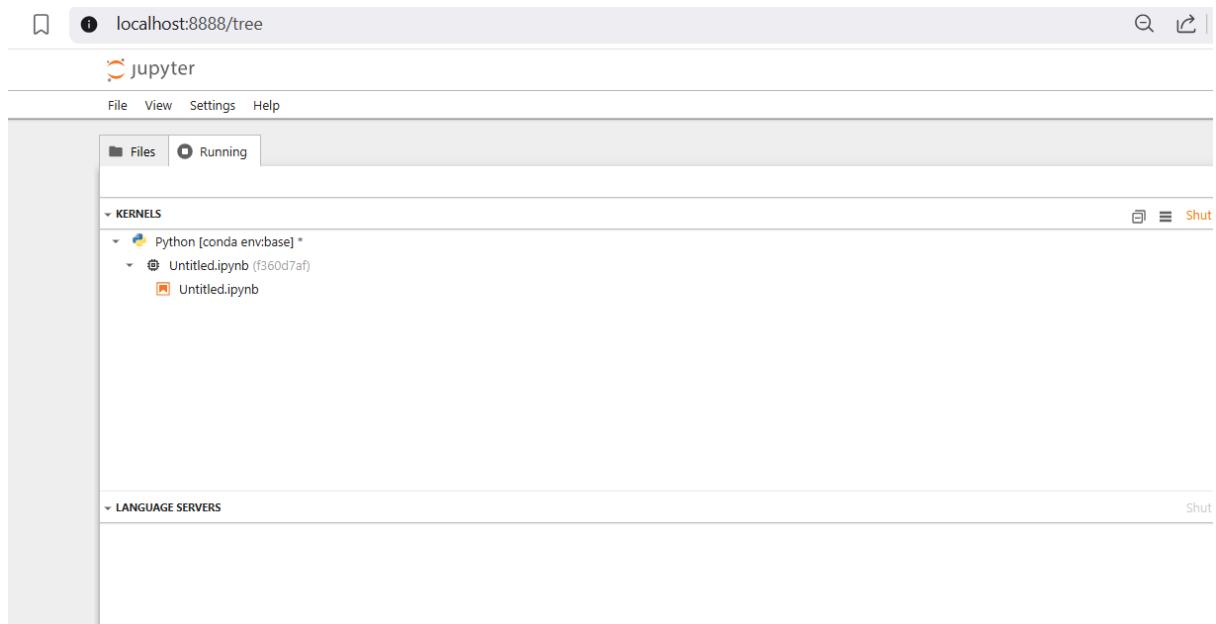


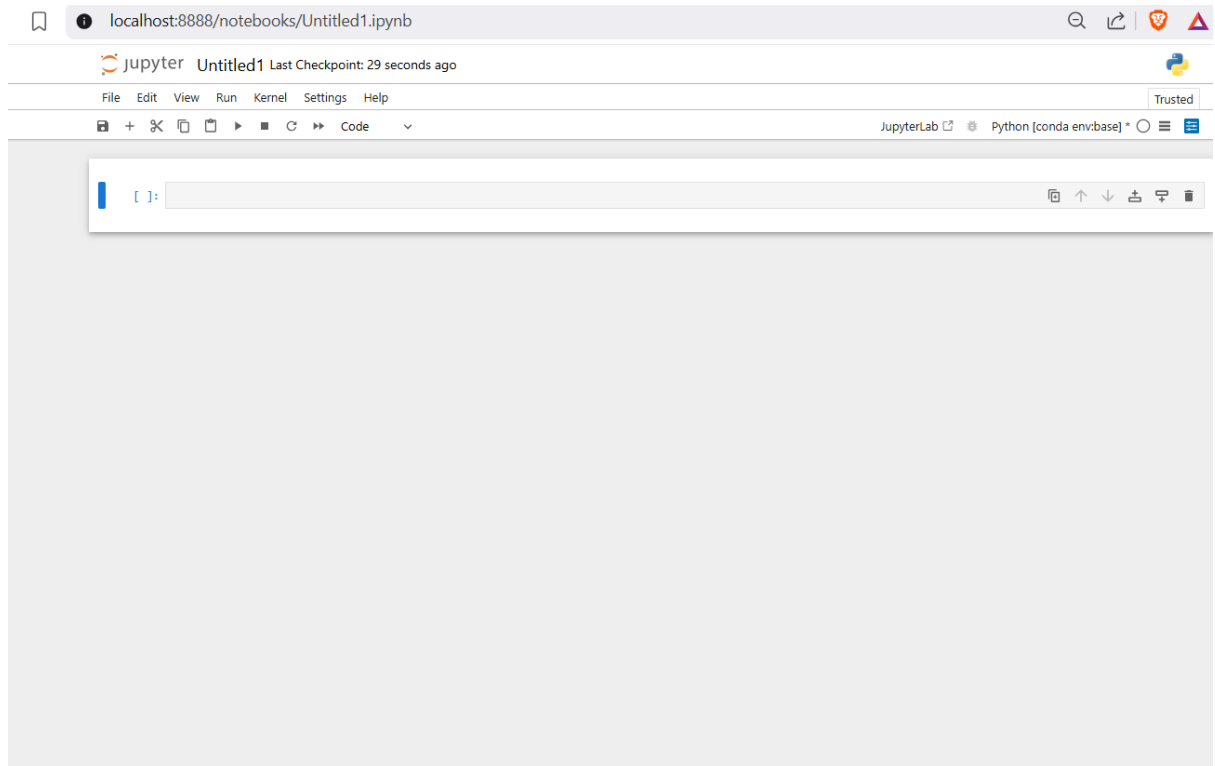
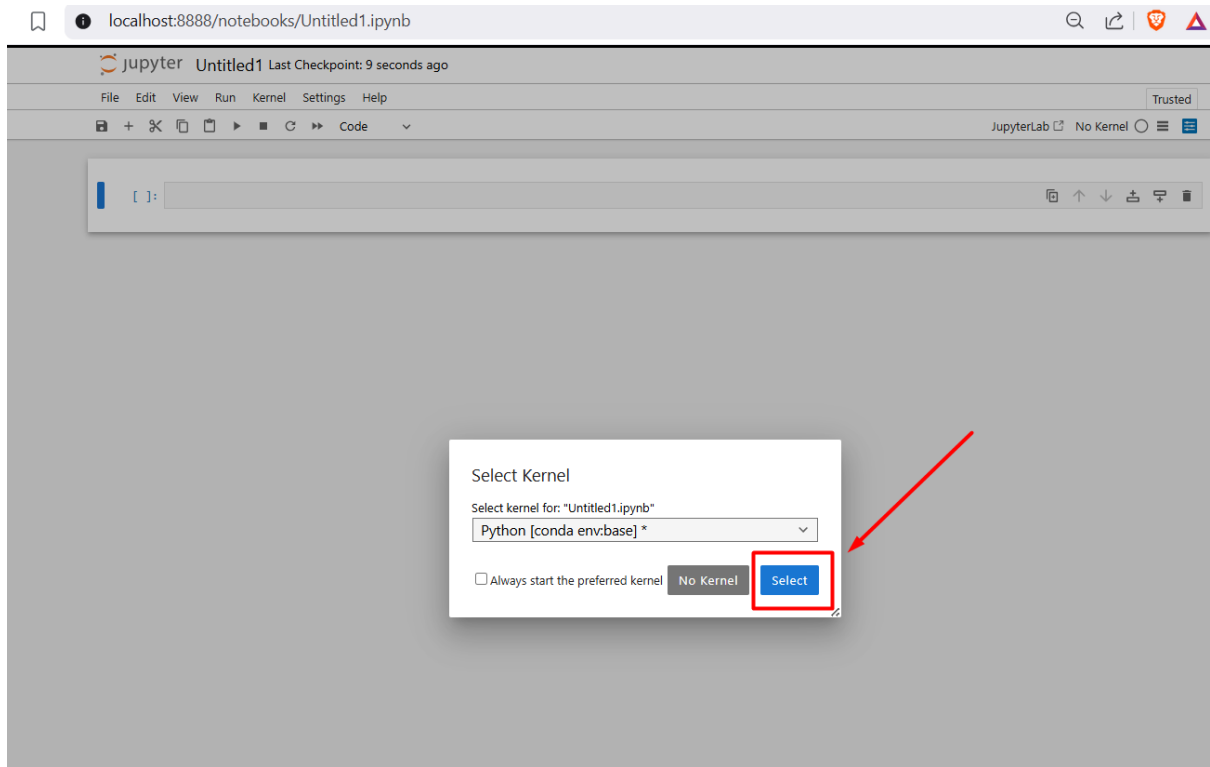


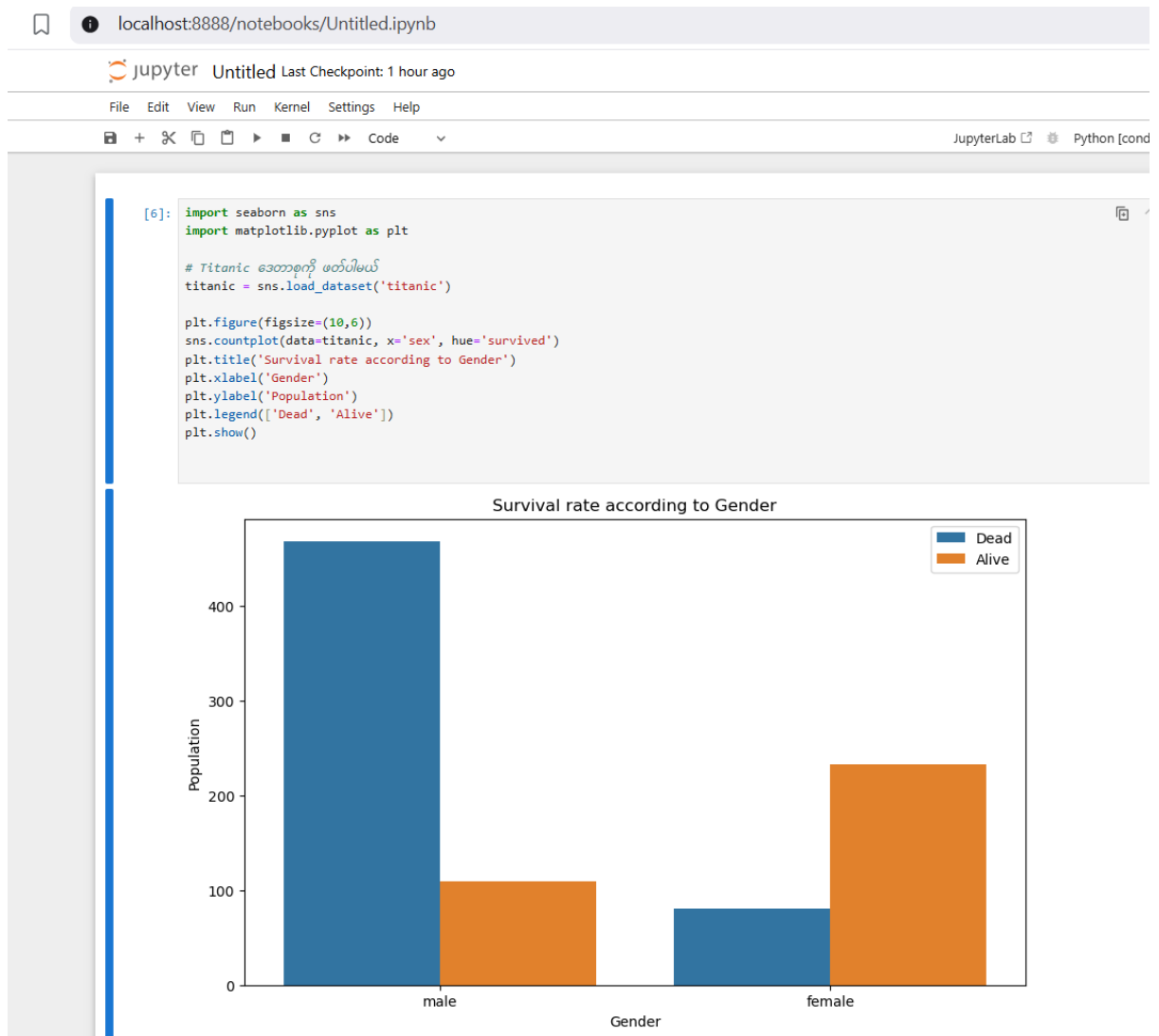
Launch



Installer



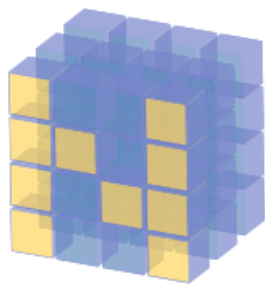






အခန်း(၁၇)

Numpy



NumPy

□ Numpy

NumPy (Numerical Python) သည် Python programming language အတွက် အရေးကြီးသော library တစ်ခုဖြစ်ပါတယ်။ NumPy ကို scientific computing, data analysis, machine learning စတဲ့နယ်ပယ်တွေမှာ အများဆုံးအသုံးပြုကြပါတယ်။ NumPy ရဲ့ အဓိက feature ကတော့ multidimensional arrays နဲ့ matrices တွေကို efficiently ကိုင်တွယ်နိုင်တာ ဖြစ်ပါတယ်။

⇒ NumPy ကို step by step ရှင်းပြပေးသွားမှာဖြစ်ပါတယ်။

၁၇.၁။ NumPy ကို Install လုပ်ခြင်း

NumPy ကို install မလုပ်ရသေးရင် အောက်ပါ command ကို အသုံးပြုပြီး install လုပ်နိုင်ပါတယ်။

Command:

```
pip install numpy
```

၁၇.၂။ NumPy ကို Import လုပ်ခြင်း

NumPy ကို အသုံးပြုဖို့အတွက် ပထမဆုံး import လုပ်ရပါမယ်။

Code:

```
import numpy as np
```

၁၇.၃။ NumPy Array ဖန်တီးခြင်း

NumPy မှာ array ဖန်တီးဖို့အတွက် `np.array()` function ကို အသုံးပြုပါတယ်။ code တွေကို ကူးပြီး run ကြည့်ပါ။

Code:

```
import numpy as np
```

```
# 1D array
```

```
arr1d = np.array([1, 2, 3, 4, 5])
```

```
print("1D Array:", arr1d)
```

```
# 2D array
```

```
arr2d = np.array([[1, 2, 3], [4, 5, 6]])
```

```
print("2D Array:\n", arr2d)
```

Output:

```
1D Array: [1 2 3 4 5]
```

```
2D Array:
```

```
[[1 2 3]
```

```
[4 5 6]]
```

၁၇.၄။ Array Attributes

NumPy array တွေရဲ့ attributes တွေကို အောက်ပါအတိုင်း စစ်ဆေးနိုင်ပါတယ်။

Code:

```
import numpy as np
```

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
```

```
print("Shape:", arr.shape) # Array dimensions
```

```
print("Size:", arr.size) # Total number of elements
```

```
print("Data Type:", arr.dtype) # Data type of elements
```

```
print("Number of Dimensions:", arr.ndim) # Number of dimensions
```

Output:

```
Shape: (2, 3)
Size: 6
Data Type: int64
Number of Dimensions: 2
```

၁၇.၅။ Special Arrays

NumPy မှာ special arrays တွေကို ဖန်တီးဖို့အတွက် functions တွေရှိပါတယ်။

Code:

```
import numpy as np

# Zeros array
zeros_arr = np.zeros((2, 3))
print("Zeros Array:\n", zeros_arr)

# Ones array
ones_arr = np.ones((2, 3))
print("Ones Array:\n", ones_arr)

# Identity matrix
identity_arr = np.eye(3)
print("Identity Matrix:\n", identity_arr)

# Array with a range of values
range_arr = np.arange(0, 10, 2)
print("Range Array:", range_arr)

# Array with evenly spaced values
linspace_arr = np.linspace(0, 1, 5)
print("Linspace Array:", linspace_arr)
```

Output:

```
Zeros Array:  
[[0. 0. 0.]  
 [0. 0. 0.]  
 Ones Array:  
 [[1. 1. 1.]  
  [1. 1. 1.]  
 Identity Matrix:  
 [[1. 0. 0.]  
  [0. 1. 0.]  
  [0. 0. 1.]  
 Range Array: [0 2 4 6 8]  
 Linspace Array: [0. 0.25 0.5 0.75 1. ]
```

၁၇.၆။ Array Indexing and Slicing

NumPy array တွေကို indexing နဲ့ slicing လုပ်နိုင်ပါတယ်။

Code:

```
import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5])  
  
# Indexing  
print("First element:", arr[0])  
print("Last element:", arr[-1])  
  
# Slicing  
print("Slice from index 1 to 3:", arr[1:4])  
print("Slice with step 2:", arr[::2])
```

**Output:**

```
First element: 1
Last element: 5
Slice from index 1 to 3: [2 3 4]
Slice with step 2: [1 3 5]
```

၁၇.၇။ Array Reshaping

Array ရဲ့ shape ကို ပြောင်းလဲဖို့အတွက် reshape() function ကို အသုံးပြုပါတယ်။

Code:

```
import numpy as np

arr = np.arange(1, 7)
reshaped_arr = arr.reshape(2, 3)
print("Reshaped Array:\n", reshaped_arr)
```

Output:

```
Reshaped Array:
[[1 2 3]
 [4 5 6]]
```

၁၇.၈။ Array Operations

NumPy array တွေမှာ mathematical operations တွေကို element-wise လုပ်နိုင်ပါတယ်။



Code:

```
import numpy as np

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Addition
print("Addition:", arr1 + arr2)

# Subtraction
print("Subtraction:", arr1 - arr2)

# Multiplication
print("Multiplication:", arr1 * arr2)

# Division
print("Division:", arr1 / arr2)

# Dot product
print("Dot Product:", np.dot(arr1, arr2))
```

Output:

```
Addition: [5 7 9]
Subtraction: [-3 -3 -3]
Multiplication: [ 4 10 18]
Division: [0.25 0.4 0.5 ]
Dot Product: 32
```

၁၇.၉။ Array Broadcasting

NumPy မှာ broadcasting ကို အသုံးပြုပြီး different shapes ရှိတဲ့ arrays တွေကို operations လုပ်နိုင်ပါတယ်။

Code:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])
scalar = 2

# Broadcasting
result = arr + scalar
print("Broadcasting Result:\n", result)
```

Output:

```
Broadcasting Result:
[[3 4 5]
 [6 7 8]]
```

၁၇.၁၀။ Array Aggregation

NumPy မှာ array တွေကို aggregate လုပ်ဖို့အတွက် functions တွေရှိပါတယ်။

Code:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

# Sum
print("Sum:", np.sum(arr))

# Mean
print("Mean:", np.mean(arr))

# Min
print("Min:", np.min(arr))

# Max
```

```
print("Max:", np.max(arr))

# Sum along axis
print("Sum along axis 0:", np.sum(arr, axis=0))
print("Sum along axis 1:", np.sum(arr, axis=1))
```

Output:

```
Sum: 21
Mean: 3.5
Min: 1
Max: 6
Sum along axis 0: [5 7 9]
Sum along axis 1: [ 6 15]
```

၁၇.၁၁။ Array Concatenation

NumPy မှာ arrays တွေကို concatenate လုပ်ဖို့အတွက် `np.concatenate()` function ကို အသုံးပြုပါတယ်။

Code:

```
import numpy as np

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Concatenation
result = np.concatenate((arr1, arr2))
print("Concatenated Array:", result)
```

Output:

```
Concatenated Array: [1 2 3 4 5 6]
```

၁၇.၁၂။ Array Sorting

NumPy မှာ array တွေကို sort လုပ်ဖို့အတွက် np.sort() function ကို အသုံးပြုပါတယ်။

Code:

```
import numpy as np

arr = np.array([3, 1, 2, 4])

# Sorting
sorted_arr = np.sort(arr)
print("Sorted Array:", sorted_arr)
```

Output:

```
Sorted Array: [1 2 3 4]
```

၁၇.၁၃။ Array Filtering

NumPy မှာ array တွေကို filter လုပ်ဖို့အတွက် boolean indexing ကို အသုံးပြုပါတယ်။

Code:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

# Filtering
filtered_arr = arr[arr > 3]
print("Filtered Array:", filtered_arr)
```


**Output:**

```
Filtered Array: [4 5]
```

၁၇.၁၄။ Random Module in NumPy

NumPy မှာ random numbers တွေကို generate လုပ်ဖို့အတွက် np.random module ကို အသုံးပြုပါတယ်။

Code:

```
import numpy as np

# Random integer
random_int = np.random.randint(0, 10)
print("Random Integer:", random_int)

# Random array
random_arr = np.random.rand(3, 3)
print("Random Array:\n", random_arr)

# Random choice
choices = np.array([1, 2, 3, 4, 5])
random_choice = np.random.choice(choices, size=3)
print("Random Choice:", random_choice)
```

Output:

```
Random Integer: 7
Random Array:
[[0.123 0.456 0.789]
 [0.321 0.654 0.987]
 [0.111 0.222 0.333]]
Random Choice: [2 5 3]
```

၁၇.၁၅။ Saving and Loading Arrays

NumPy မှာ arrays တွေကို save နဲ့ load လုပ်ဖို့အတွက် functions တွေရှိပါတယ်။

Code:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

# Save array to file
np.save('my_array.npy', arr)

# Load array from file
loaded_arr = np.load('my_array.npy')
print("Loaded Array:", loaded_arr)
```

Output:

```
Loaded Array: [1 2 3 4 5]
```

✍ Short Notes:

NumPy သည် Python programming language အတွက် အရေးကြီးသော library တစ်ခုဖြစ်ပြီး scientific computing, data analysis, machine learning စတဲ့နယ်ပယ်တွေမှာ အများဆုံးအသုံးပြုကြပါတယ်။ NumPy ရဲ့အဓိက feature ကတော့ multidimensional arrays နဲ့ matrices တွေကို efficiently ကိုင်တွယ်နိုင်တာပါ။ NumPy ကို အသုံးပြုပြီး မတူညီတဲ့ operations တွေကို လေ့လာပြီး လိုအပ်တဲ့အခါမှာ အသုံးပြုနိုင်ပါတယ်။



လေ့ကျင့်ရန် -

Practice 1:

```
import numpy as np
```

```
#1-d array
```

```
nums=np.array([1,2,3,4,5,6,7,8,9])
```

```
print("one dimensional array")
```

```
print(nums.shape)
```

```
nums=np.array([[1,2,3,4],[5,6,7,8]])
```

```
print("2 dimensional array")
```

```
print(nums.shape)
```

```
nums=np.array([ [1,2,3,4],[5,6,7,8],[9,10,11,12]] ,  
[[21,22,23,24],[25,26,27,28],[29,30,31,32]] )
```

```
print("3 dimensional array")
```

```
print(nums.shape)
```

Output:

```
One dimensional array
(9,)
2 dimensional array
(2,4)
3 dimensional array
(2,3,4)
```

Practice 2:

#Shape and Reshape Example

```
import numpy as np
```

```
nums=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
newArr=nums.reshape(3,4)
print('Reshape 1 D array to 2 D Array')
print(newArr)
```

```
nums=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
newArr=nums.reshape(3,2,2)
print('Reshape 1 D array to 3 D Array')
print(newArr)
```

```
nums=np.array([ [1,2],[3,4]] , [[5,6],[7,8]] , [[9,10],[11,12]] )
```

```
newArr=nums.reshape(-1)
```



```
print('Reshape 3-D to 1-D array')
print(newArr)

nums=np.array([ [[1,2],[3,4]] , [[5,6],[7,8]] , [[9,10],[11,12]] ])

newArr=nums.reshape(2,2,-1)
print('Reshape 3-D to 2-D array')
print(newArr)

nums=np.array( [[1,2,3,4],[5,6,7,8]] )

newArr=nums.reshape(-1)
print('Reshape 2-D to 1-D array')
print(newArr)
```

Output:

Reshape 1 D array to 2D array

```
[[1  2  3  4]
 [5  6  7  8]
 [ 9 10 11 12]]
```

Reshape 1D to 3D array

```
[[[ 1  2]
 [ 3  4]]
 [[ 5  6]
 [ 7  8]]
 [[ 9 10]
 [11 12]]]
```

Reshape 3D to 1D

```
[1  2  3  4  5  6  7  8  9 10 11
 12]
```

Reshape 3D to 2D

```
[[[ 1  2  3]
 [ 4  5  6]]
 [[ 7  8  9]
 [10 11 12]]]
```

Reshape 2D to 1D Array

```
[1  2  3  4  5  6  7  8]
```

Practice 3:

#Indexing

```
import numpy as np
```

```
nums=np.array([41,22,33,44,15,16,17,8,9])
newArr=nums[1:3]
print('print nums Index between 1 and 3 ')
print(newArr)
```

```
nums=np.array([41,22,33,44,15,16,17,8,9])
print('print nums Index from 0 to 5 ')
newArr=nums[:5]
print(newArr)
```

```
nums=np.array([41,22,33,44,15,16,17,8,9])
print('print nums Index -3 to -1 ')
newArr=nums[-3:-1]
print(newArr)
```

```
nums=np.array([41,22,33,44,15,16,17,8,9])
print('print nums Index -3 to end ')
newArr=nums[-3:]
print(newArr)
```



Output:

Print nums index between 1 and 3

[22 33]

Print nums index from 0 to 5

[41 22 33 44 15]

Print nums index -3 to -1

[17 8]

Print nums index -3 to end

[17 8 9]

Practice 4:

```
import numpy as np
```

```
obj_array=[  
    {'name':'apple','price':2000},  
    {'name':'orange','price':1500},  
    {'name': 'honeydue', 'price': 5000},  
    {'name': 'lime', 'price': 200},  
    {'name':'mango','price':1000},  
    {'name':'banana','price':2500},  
    {'name': 'watermelon', 'price': 3000},  
    {'name': 'lychee', 'price': 4500}  
]
```




```
items=np.array(obj_array)
filter_arr=[]
for item in items:
    if item['price']>=2000:
        filter_arr.append(True)
    else:
        filter_arr.append(False)

print('Item List item price is greater than 2000 ')
item_list=items[filter_arr]
print(item_list)
```

Output:

```
Item List item price is greater than 2000
[{'name' : 'apple' , 'price' :2000},{ 'name' : 'honeydue' , 'price' :5000}
{'name' : 'banana' , 'price' :2500},{ 'name' : 'watermelon' , 'price' :3000}
{'name' : 'lychee' , 'price' :4500}
]
```



အခန်း(၁၈)

Pandas



▣ Pandas

pandas ဆိုတာက data analysis နဲ့ data manipulation လုပ်ဖို့အတွက် အသုံးဝင်တဲ့ library တစ်ခုဖြစ်ပါတယ်။ pandas ကို အဓိကအားဖြင့် structured data (ဥပမာ- table, spreadsheet) တွေကို လွယ်ကူစွာ စီမံခန့်ခွဲဖို့အတွက် အသုံးပြုပါတယ်။ pandas မှာ DataFrame နဲ့ Series ဆိုတဲ့ data structure အဓိက ၂ မျိုးရှိပါတယ်။

၁၈.၁။ pandas ကို ဘယ်လိုအသုံးပြုမလဲ?

pandas ကို အသုံးပြုဖို့အတွက် ပထမဆုံး library ကို install လုပ်ရပါမယ်။ ပြီးရင် import လုပ်ပြီး အသုံးပြုနိုင်ပါတယ်။

```
python
```

```
Copy
```

```
# pandas ကို install လုပ်ခြင်း (အကယ်၍ မလုပ်ရသေးရင်)
```

```
!pip install pandas
```

```
# pandas ကို import လုပ်ခြင်း
```

```
import pandas as pd
```

၁၈.၂။ DataFrame ဆိုတာဘာလဲ?

DataFrame ဆိုတာက table ပုံစံ data တွေကို ကိုယ်စားပြုတဲ့ data structure တစ်ခုဖြစ်ပါတယ်။ DataFrame မှာ rows (အတန်းများ) နဲ့ columns (ကော်လံများ) ရှိပါတယ်။

Example: DataFrame ဖန်တီးခြင်း

```
import pandas as pd

# Dictionary ကို အသုံးပြုပြီး DataFrame ဖန်တီးခြင်း
data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "City": ["New York", "Los Angeles", "Chicago"]
}

df = pd.DataFrame(data)
print(df)
```

Output:

	Name	Age	City
0	Alice	25	New York
1	Bob	30	Los Angeles
2	Charlie	35	Chicago

၁၈.၃။ Series ဆိုတာဘာလဲ?

Series ဆိုတာက DataFrame ရဲ့ တစ်ခုတည်းသော column တစ်ခုကို ကိုယ်စားပြုတဲ့ data structure တစ်ခုဖြစ်ပါတယ်။ Series မှာ index နဲ့ values တွေပါဝင်ပါတယ်။

နမူနာ: Series ဖန်တီးခြင်း

```
import pandas as pd
```

```
# Series ဖန်တီးခြင်း
```

```
ages = pd.Series([25, 30, 35], name="Age")
```

```
print(ages)
```

Output:

```
0    25
```

```
1    30
```

```
2    35
```

```
Name: Age, dtype: int64
```

၁၈.၄။ pandas ကို အသုံးပြုပြီး data တွေကို ဘယ်လိုဖတ်မလဲ?

pandas ကို အသုံးပြုပြီး CSV, Excel, SQL database စတဲ့ file format တွေကနေ data တွေကို ဖတ်နိုင်ပါတယ်။

နမူနာ: CSV file ဖတ်ခြင်း

```
import pandas as pd

# CSV file ဖတ်ခြင်း
df = pd.read_csv("data.csv")
print(df)
```

၁၈.၅။ DataFrame မှာ data တွေကို ဘယ်လိုရွေးချယ်မလဲ?

DataFrame ထဲက data တွေကို column နဲ့ row တွေကို အသုံးပြုပြီး ရွေးချယ်နိုင်ပါတယ်။

နမူနာ: Column ရွေးချယ်ခြင်း

```
# Name column ကို ရွေးချယ်ခြင်း
names = df["Name"]
print(names)
```

နမူနာ: Row ရွေးချယ်ခြင်း

```
# ပထမဆုံး row ကို ရွေးချယ်ခြင်း
first_row = df.iloc[0]
print(first_row)
```

၁၈.၆။ DataFrame မှာ data တွေကို ဘယ်လိုစစ်ထုတ်မလဲ?

Conditional filtering ကို အသုံးပြုပြီး data တွေကို စစ်ထုတ်နိုင်ပါတယ်။

နမူနာ: Age က 30 ထက်ကြီးတဲ့ row တွေကို စစ်ထုတ်ခြင်း

Code:

```
filtered_df = df[df["Age"] > 30]
print(filtered_df)
```

၁၈.၇။ DataFrame မှာ data တွေကို ဘယ်လိုပြုပြင်မလဲ?

DataFrame ထဲက data တွေကို update လုပ်တာ၊ ဖျက်တာ၊ ထပ်ထည့်တာတွေ လုပ်နိုင်ပါတယ်။

နမူနာ: Column တစ်ခုထပ်ထည့်ခြင်း

Code:

```
# New column ထပ်ထည့်ခြင်း
df["Salary"] = [50000, 60000, 70000]
print(df)
```

နမူနာ: Row တစ်ခုထပ်ထည့်ခြင်း

Code:

```
# New row ထပ်ထည့်ခြင်း
new_row = {"Name": "David", "Age": 40, "City": "Houston"}
df = df.append(new_row, ignore_index=True)
print(df)
```

၁၈.၈။ pandas ကို အသုံးပြုပြီး data တွေကို ဘယ်လိုသိမ်းမလဲ?

pandas ကို အသုံးပြုပြီး data တွေကို CSV, Excel စတဲ့ format တွေနဲ့ သိမ်းဆည်းနိုင်ပါတယ်။

နမူနာ: CSV file အဖြစ် သိမ်းဆည်းခြင်း

Code:

```
df.to_csv("output.csv", index=False)
```

၁၈.၉။ pandas ရဲ့ အားသာချက်များ

- Data manipulation လုပ်ရာမှာ လွယ်ကူပြီး မြန်ဆန်ပါတယ်။
- Large datasets တွေကို ထိရောက်စွာ စီမံခန့်ခွဲနိုင်ပါတယ်။
- Data cleaning, transformation, analysis တွေကို လွယ်ကူစွာ လုပ်ဆောင်နိုင်ပါတယ်။

၁၈.၁၀။ နမူနာ Program

Code:

```
import pandas as pd
```

```
# Data ဖန်တီးခြင်း
```

```
data = {  
    "Name": ["Alice", "Bob", "Charlie"],  
    "Age": [25, 30, 35],  
    "City": ["New York", "Los Angeles", "Chicago"]  
}
```



```
# DataFrame ဖန်တီးခြင်း
df = pd.DataFrame(data)

# Data ကို ကြည့်ရှုခြင်း
print("Original DataFrame:")
print(df)

# Column ရွေးချယ်ခြင်း
print("\nName Column:")
print(df["Name"])

# Row ရွေးချယ်ခြင်း
print("\nFirst Row:")
print(df.iloc[0])

# Data စစ်ထုတ်ခြင်း
print("\nAge > 30:")
print(df[df["Age"] > 30])

# New column ထပ်ထည့်ခြင်း
df["Salary"] = [50000, 60000, 70000]
print("\nDataFrame with Salary Column:")
print(df)

# CSV file အဖြစ် သိမ်းဆည်းခြင်း
df.to_csv("output.csv", index=False)
print("\nData saved to output.csv")
```

ဒီနမူနာတွေကို ကြည့်ပြီး pandas ကို အသုံးပြုပြီး data analysis လုပ်နည်းတွေကို နားလည်သွားမယ်လို့ မျှော်လင့်ပါတယ်။



လေ့ကျင့်ရန် -

Practice-1:

Data.csv

Duration,Date,Pulse,Maxpulse,Calories

```
60,'2024/02/12',110,130,409.1
60,'2024/02/13',117,145,
600,'2024/02/14',103,135,340
45,'2024/02/15',109,175,282.4
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
60,20240217,102,127,
```



```
60,'2024/02/18',110,136,374
450,'2024/02/19',104,134,253.3
30,'2024/02/20',109,133,195.1
60,'2024/02/21',98,124,269
250,'2024/02/22',103,147,329.3
60,'2024/02/23',100,120,250.7
60,'2024/02/24',106,128,345.3
```

Code:

```
import pandas as pd

#read file
df=pd.read_csv('data.csv')
#show column names and their data types
print(df.dtypes)

#show statistics
print(df.describe())

#show top 5 rows
print(df.head())

#show top 20 rows
print(df.head(20))

#show bottom 5 rows
print(df.tail())

#set max rows
pd.set_option('display.max_rows',500)
```



```
print(df)
```

output:

```
#print(df.dtypes)
```

Duration	int64
Date	Object
Pulse	int64
Maxpulse	int64
Calories	float64
Dtype	object

```
#print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype  
---  -
0   Duration    22 non-null    int64   
1   Date        22 non-null    object  
2   Pulse       22 non-null    int64   
3   Maxpulse    22 non-null    int64   
4   Calories    20 non-null    float64  
dtypes: float64(1), int64(3), object(1)
memory usage: 1012.0+ bytes
None
```

```
#print(df.describe())
```

Output:

```
C:\Users\Dell\PycharmProjects\basic_lessons\.venv\Scripts\python
      Duration      Pulse  Maxpulse  Calories
count  22.000000  22.000000  22.000000  20.000000
mean   102.045455  110.954545  141.545455  355.410000
std    145.452161    6.827656   12.113461   68.721474
min     30.000000   98.000000  120.000000  195.100000
25%     45.000000  104.500000  133.250000  317.575000
50%     45.000000  113.500000  147.500000  406.000000
75%     60.000000  117.000000  148.000000  406.000000
max     600.000000  117.000000  175.000000  409.100000

Process finished with exit code 0
```



#display top 5 rows - default

Print(df.head())

Output:

Duration	Date	Pulse	Maxpulse	Calories
60	'2024/02/12'	110	130	409.1
60	'2024/02/13'	117	145	NaN
600	'2024/02/14'	103	135	340.0
45	'2024/02/15'	109	175	282.4
45	'2024/02/16'	117	148	406.0

#display top 20 rows – customize

Print(df.head(20))

Output:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3



```
#display last row 5 – default
```

```
Print(df.tail())
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

```
#set maximum rows
```

```
pd.set_option('display.max_rows',500)
```

```
print(df)
```

Output:



	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

Practice-2:

#Removing missing values (NaN)

#command: dropna()

import pandas as pd

#read file

df=pd.read_csv('data.csv')

pd.set_option('display.max_rows',200)

#new_df=df.dropna()

print(df.to_string())

Output:

	Duration	Date	Pulse	Maxpulse	Calories
	60	'2024/02/12'	110	130	409.1
	60	'2024/02/13'	117	145	NaN
	600	'2024/02/14'	103	135	340.0
	45	'2024/02/15'	109	175	282.4
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
0	45	'2024/02/16'	117	148	406.0
1	45	'2024/02/16'	117	148	406.0
2	45	'2024/02/16'	117	148	406.0
3	45	'2024/02/16'	117	148	406.0
4	60	20240217	102	127	NaN
5	60	'2024/02/18'	110	136	374.0
6	450	'2024/02/19'	104	134	253.3
7	30	'2024/02/20'	109	133	195.1
8	60	'2024/02/21'	98	124	269.0
9	250	'2024/02/22'	103	147	329.3
0	60	'2024/02/23'	100	120	250.7
1	60	'2024/02/24'	106	128	345.3

#After removing missing values

```
import pandas as pd
```

```
#read file
```

```
df=pd.read_csv('data.csv')
```

```
pd.set_option('display.max_rows',200)
```

```
new_df=df.dropna()
```

```
print(new_df.to_string())
```



	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

Practice-3:

#Original Data Set

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
print(df)
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

#After Removing Duplicated Data

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
# remove opeation
```

```
df.drop_duplicates(inplace=True)
```

```
#after removing duplicates
```

```
print(df)
```

Output:



	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

Practice-4:

#Original Data with Wrong Data

```
C:\Users\Dell\PycharmProjects\basic_lessons\.venv\Scripts\python.exe C:\Users\Dell\Py
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	545	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	300	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1



#After Correcting Wrong Data

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
for x in df.index:
```

```
    if df.loc[x,'Duration']>60:
```

```
        df.loc[x,'Duration']=60
```

```
print(df)
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	60	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	60	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	60	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	60	'2024/02/19'	104	134	253.3

Practice-5:

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
print(df.info())
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   Duration    22 non-null     int64
 1   Date        22 non-null     object
 2   Pulse       22 non-null     int64
 3   Maxpulse    22 non-null     int64
 4   Calories    20 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1012.0+ bytes
None
```

#After Converting Date Data type to Date Time Data type



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Duration    22 non-null    int64
1   Date        21 non-null    datetime64[ns]
2   Pulse       22 non-null    int64
3   Maxpulse    22 non-null    int64
4   Calories    20 non-null    float64
dtypes: datetime64[ns](1), float64(1), int64(3)
memory usage: 1012.0 bytes
None

Process finished with exit code 0
```



အခန်း(၁၉)

Matplotlib



□ Matplotlib

Matplotlib သည် Python programming language အတွက် အသုံးအများဆုံး data visualization library တစ်ခုဖြစ်ပါတယ်။ ဒီ library ကို scientific computing, data analysis နဲ့ engineering fields တွေမှာ အထူးသဖြင့် အသုံးများပါတယ်။ လွယ်ကူသော graph များ ရေးဆွဲနိုင်ခြင်း - Line plots, bar charts, histograms, scatter plots စသည်တို့ကို ရိုးရှင်းစွာ ဖန်တီးနိုင်ပါတယ်။ High-quality output - Publication-quality figures များကို ဖန်တီးနိုင်ပါတယ်။ Customization အပြည့်အစုံ - ဂရပ်တစ်ခုလုံး၏ အသေးစိတ်အချက်အလက်များကို စိတ်ကြိုက်ပြင်ဆင်နိုင်ပါတယ်။ Interactive features - Zooming, panning, updating စသည့် interactive features များ ပါဝင်ပါတယ်။

၁၉.၁။ ရိုးရှင်းသော ဥပမာ

```
import matplotlib.pyplot as plt
```

```
# Data ပြင်ဆင်ခြင်း
```

```
x = [1, 2, 3, 4, 5]
```

```
y = [2, 4, 6, 8, 10]
```

```
# Line plot ရေးဆွဲခြင်း
```

```
plt.plot(x, y)
```

```
# Labels နှင့် title ထည့်ခြင်း
```

```
plt.xlabel('X Axis')
```

```
plt.ylabel('Y Axis')
```

```
plt.title('ရိုးရှင်းသော Line Plot')
```

```
# Graph ပြသခြင်း
```

```
plt.show()
```

၁၉.၂။ Matplotlib ၏ အဓိက Components များ

1. **Figure** - ဂရပ်အားလုံးပါဝင်သော အထုပ်ကြီးတစ်ခု
2. **Axes** - တစ်ခုတည်းသော plot (တစ်ခု figure ထဲတွင် axes အများအပြားပါနိုင်ပါတယ်)
3. **Axis** - x-axis နှင့် y-axis များ
4. **Artist** - Figure ပေါ်ရှိ text, lines, rectangles စသည့် အရာအားလုံး

Matplotlib သည် Python တွင် data visualization လုပ်ရန် အစွမ်းထက်သော tool တစ်ခုဖြစ်ပြီး သင်ယူရလွယ်ကူကာ လိုအပ်သလို customize လုပ်နိုင်ပါတယ်။

Example 1

```
import matplotlib.pyplot as plt
import numpy as np

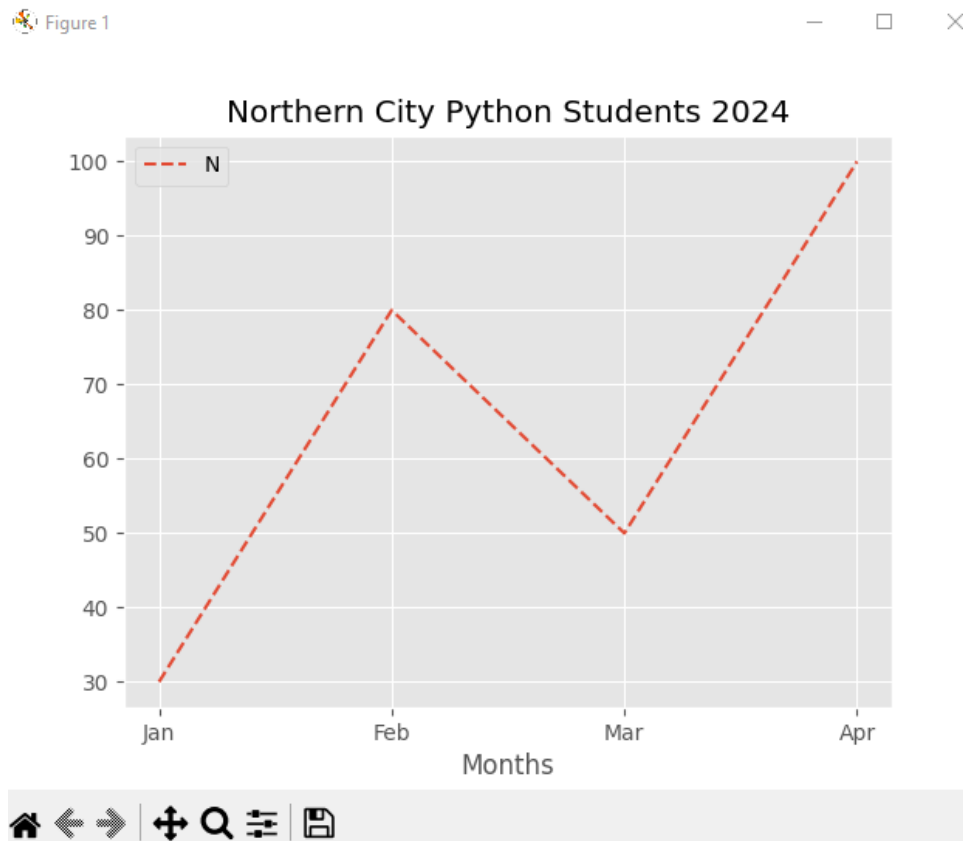
#print(plt.style.available)
plt.style.use('ggplot')
months = ['Jan','Feb','Mar','Apr']
qty = [30, 80, 50, 100]

plt.plot(months,qty, linestyle = 'dashed')
plt.title('Northern City Python Students 2024')

plt.xlabel("Months")
plt.legend("No of Students")
```

```
#plt.savefig('line_graph.png')  
plt.show()
```

Output:



Example 2

```
#Myanmar Cities and their population
```

```
from matplotlib import pyplot as plt
```



```
dev_x=['Yangon','Mandalay','Mawlamyine','Taunggyi','Lashio','Myitkyina']
dev_y=[6200000,2500000,1600000,550000,210000,870000];

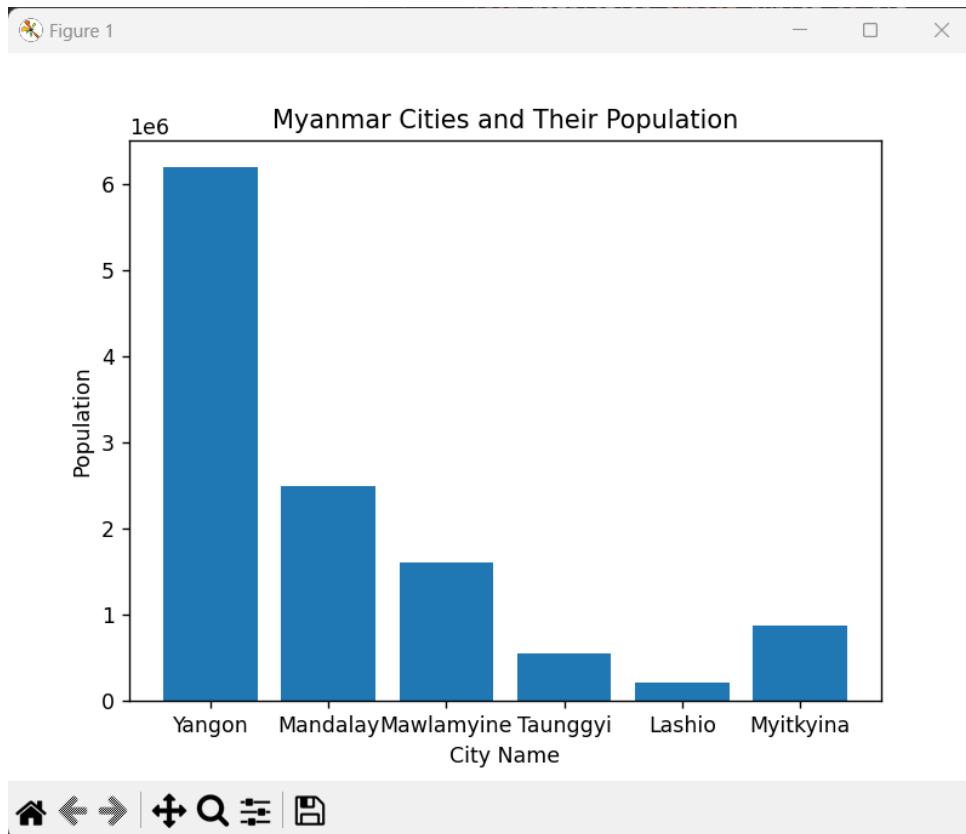
#setting bar graph
plt.bar(dev_x,dev_y)

#describing labels of the graph
plt.xlabel('City Name')
plt.ylabel('Population')

#describing graph title
plt.title('Myanmar Cities and Their Population')

#show graph
plt.show()
```

Output:



Example 3

#2024 Northern City enrolled Programming Student

#According to Months and Populations

import numpy as np

from matplotlib import pyplot as plt

Setting the width of the bars

bar_width = 0.25

x=np.arange(5)

Calculating bar positions for both groups

bar_positions1 = np.arange(5)

bar_positions2 = bar_positions1 + bar_width

bar_positions3 = bar_positions2 + bar_width

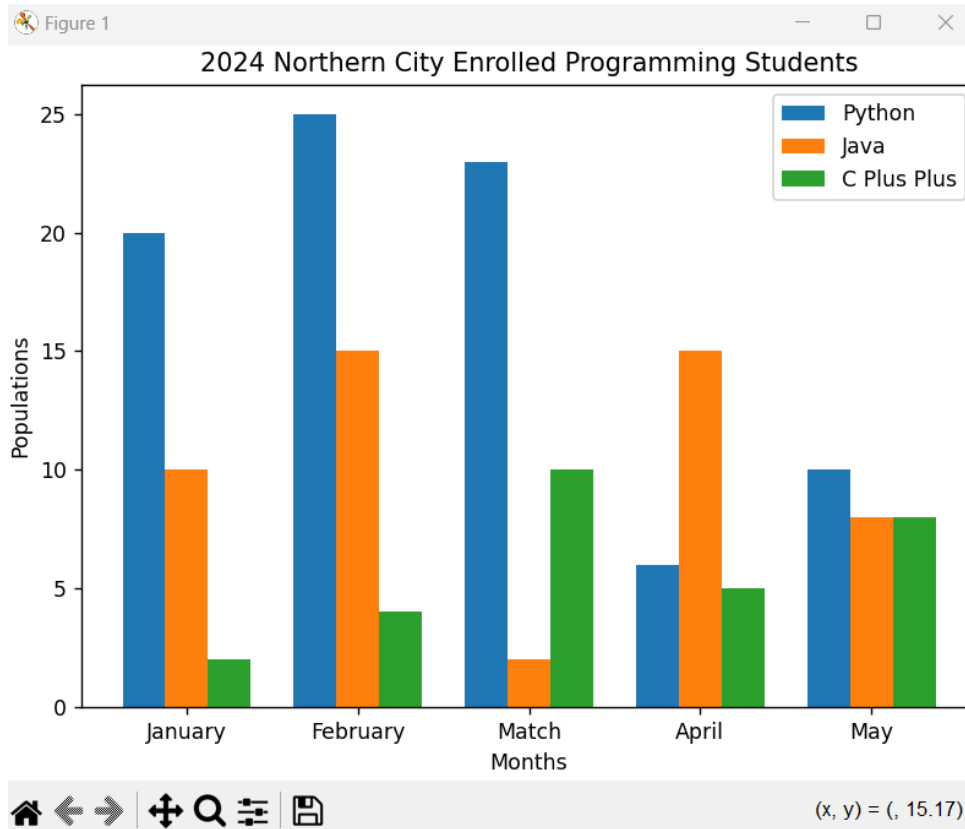


```
ax = plt.subplots(layout='constrained')
dev_x=('January','February','March','April','May')
dev_y_python=[20,25,23,6,10];
plt.bar(bar_positions1,dev_y_python,width=bar_width,label='Python')
dev_y_java=[10,15,2,15,8];
plt.bar(bar_positions2,dev_y_java,width=bar_width,label='Java')
dev_y_cpp=[2,4,10,5,8];
plt.bar(bar_positions3,dev_y_cpp,width=bar_width,label='C Plus Plus')

plt.legend(['Python','Java','C Plus Plus'])
plt.xlabel('Months')
## plt.xticks(range(len(list)),col_names)
plt.xticks(x+bar_width,dev_x)
plt.ylabel('Populations')
plt.title('2024 Northern City Enrolled Programming Students')

plt.show()
```

Output:



Example 4

#2024 April Entrolled Student List

```
import numpy as np
```

```
from matplotlib import pyplot as plt
```

```
dev_x=('Java','C#','Python','CPP','PHP')
```

```
dev_y=[20,25,23,6,10]
```

```
#sorted_y=dev_y.sort(reverse=True)
```

```
sorted_y=dev_y.sort()
```

```
plt.barh(dev_x,dev_y,data=sorted_y)
```

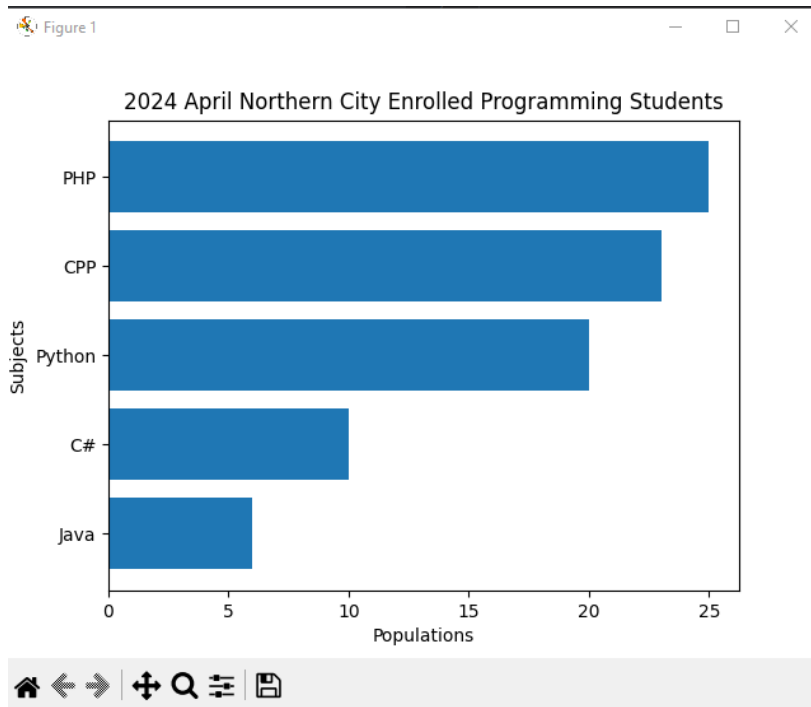
```
plt.ylabel('Subjects')
```

```
plt.xlabel('Populations')
```

```
plt.title('2024 April Northern City Enrolled Programming Students')
```

```
plt.show()
```

Output:



Example 5

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

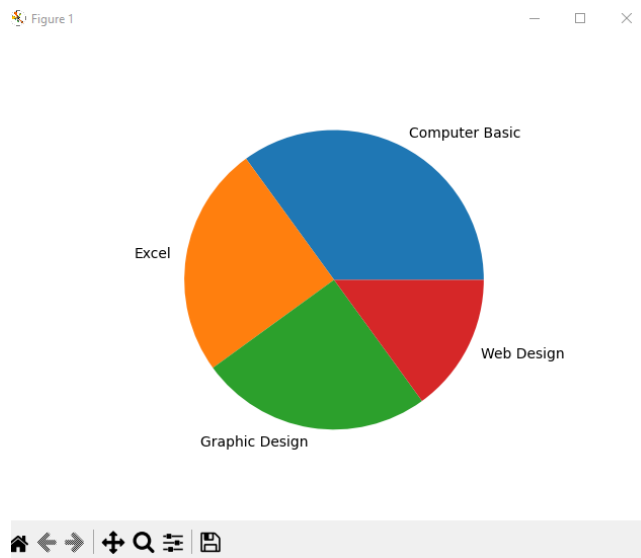
```
data = np.array([35, 25, 25, 15])
```

```
mylabels = ["Computer Basic", "Excel", "Graphic Design", "Web Design"]
```

```
plt.pie(data, labels = mylabels)
```

```
plt.show()
```


Ouput:



Example 6

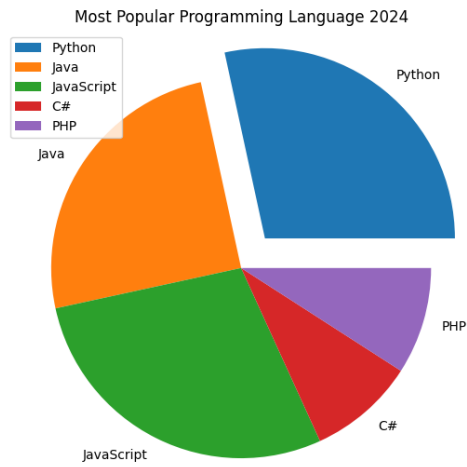
```
import matplotlib.pyplot as plt
import numpy as np

data = np.array([25, 22, 25, 8,8])
mylabels = ["Python", "Java", "JavaScript", "C#","PHP"]
myexplode = [0.2, 0, 0, 0,0]

plt.pie(data, labels = mylabels,explode=myexplode)
plt.title("Most Popular Programming Language 2024")
plt.legend(["Python", "Java", "JavaScript", "C#","PHP"])
plt.show()
```



Output:





လေ့ကျင့်ရန်-

Practice

Title: Norten City Entrolled Stuent's 2023

Dataset:

```
data = [180, 420, 250, 210, 80]
```

```
subjects = ["Python", "Java", "JavaScript", "C#", "PHP"]
```

Matplotlib Practice

Main MENU

[1] Draw Line Graph

[2] Draw Vertical Bar Graph

[3] Draw Horizontal Bar Graph

[4] Draw Pie Graph

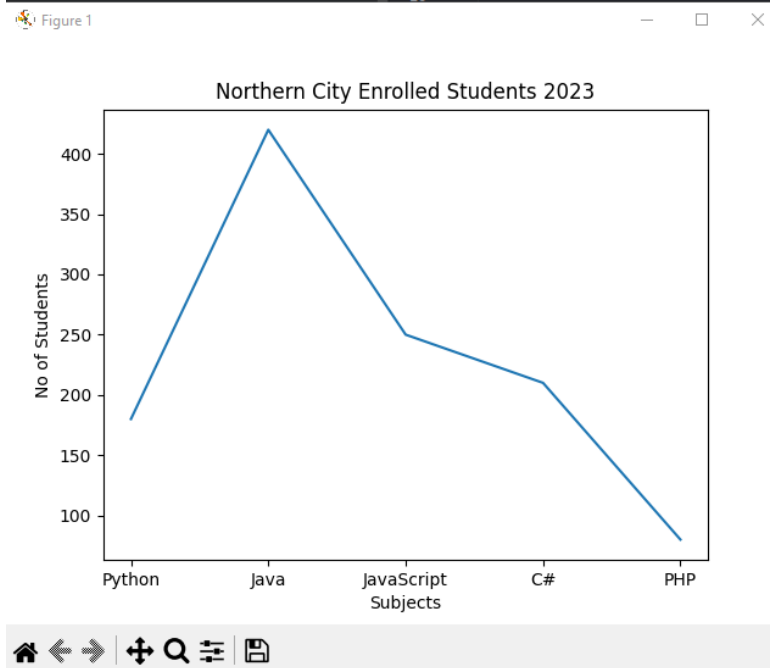
[5] Exit Program



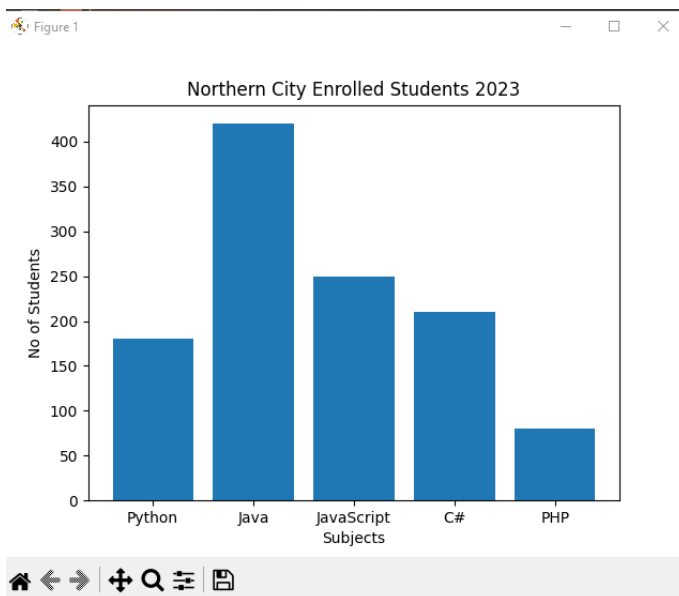
Enter choice:

Output:

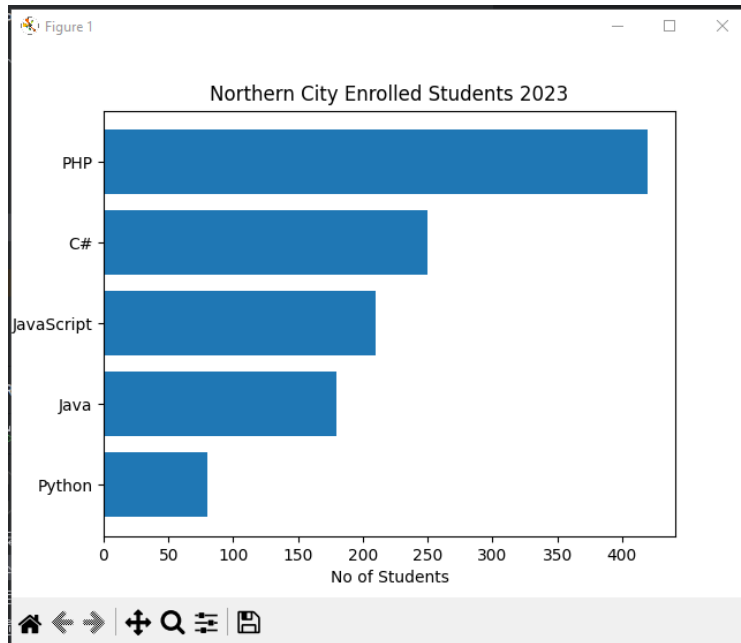
Option-1 Output:



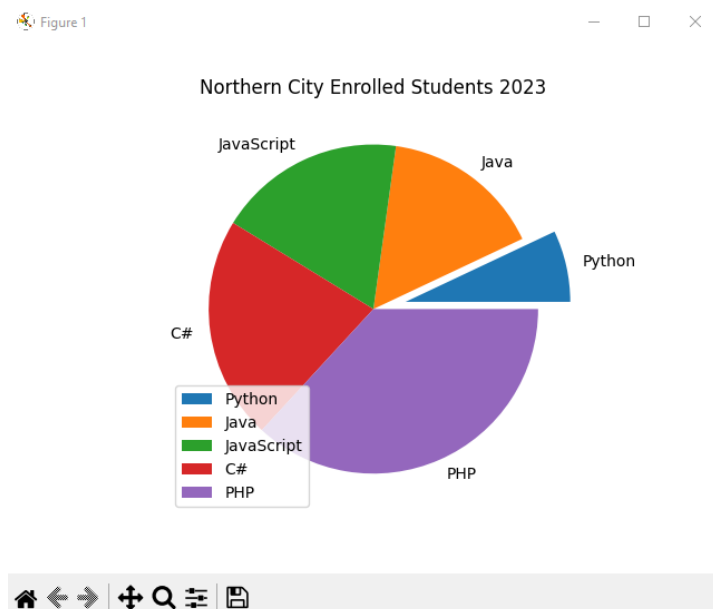
Option-2 Output:



Option-3 Output:



Option 4 – Output:





အခန်း(၂၀)

Seaborn



❑ Seaborn

Seaborn သည် Python အတွက် အဆင့်မြင့် data visualization library တစ်ခုဖြစ်ပါတယ်။ Matplotlib ကို အခြေခံပြီး ပိုမိုလှပသော နှင့် ရှုပ်ထွေးမှုနည်းသော ဂရပ်ပုံများကို ရေးဆွဲနိုင်ပါတယ်။ အလွယ်တကူအသုံးပြုနိုင်မှု - ရိုးရှင်းသော syntax ဖြင့် ရှုပ်ထွေးသော visualization များကို ဖန်တီးနိုင်ပါတယ်။ အလှအပပြည့်စုံမှု - Default theme နှင့် color palette များကြောင့် လှပသော ဂရပ်များရရှိပါတယ်။ Statistical visualization - Data distribution, regression, correlation စသည်တို့ကို အထူးပြုဖော်ပြနိုင်ပါတယ်။

၂၀.၁။ ဥပမာ Code များ

```
import seaborn as sns
import matplotlib.pyplot as plt

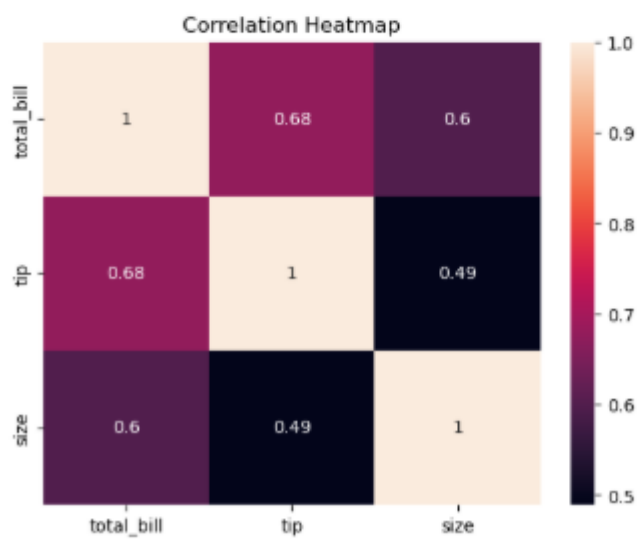
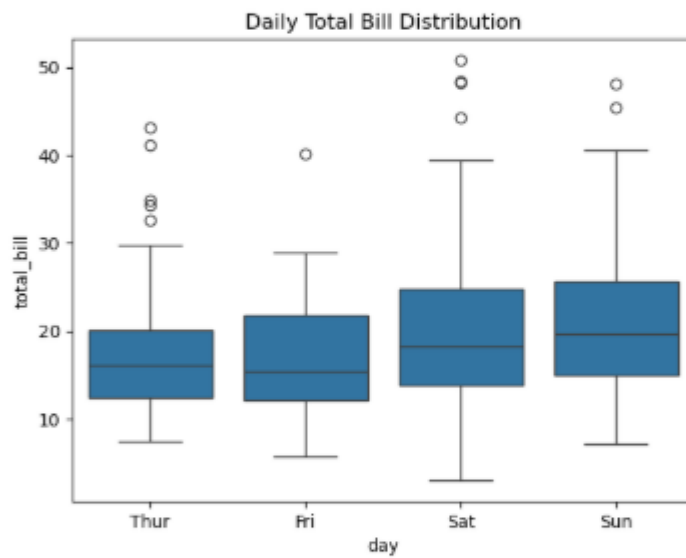
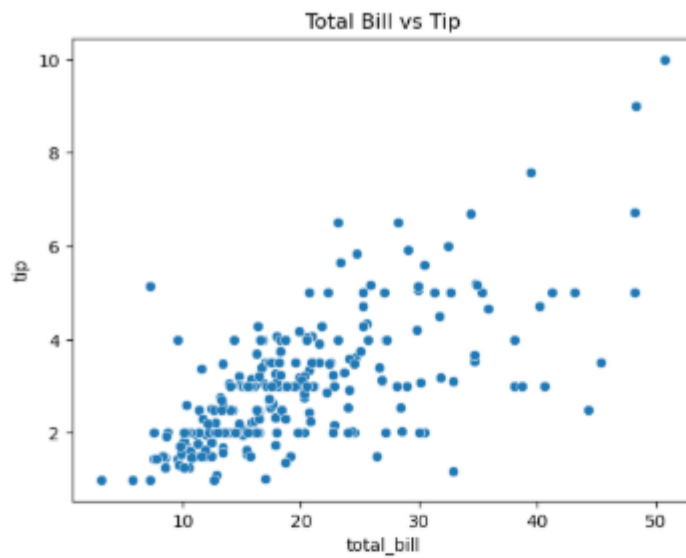
# Sample data ဖြင့် ဥပမာ
tips = sns.load_dataset("tips")

# Scatter plot
sns.scatterplot(x="total_bill", y="tip", data=tips)
plt.title("Total Bill vs Tip")
plt.show()

# Box plot
sns.boxplot(x="day", y="total_bill", data=tips)
plt.title("Daily Total Bill Distribution")
plt.show()

# Heatmap
correlation = tips.corr()
sns.heatmap(correlation, annot=True)
plt.title("Correlation Heatmap")
plt.show()
```

Output:



၂၀.၂။ Seaborn ၏ အဓိကအသုံးဝင်မှုများ

- Data exploration - ဒေတာများကို မြန်မြန်ဆန်ဆန် နားလည်ရန်
- Statistical relationships - Variable များကြားဆက်နွှယ်မှုများကို ဖော်ပြရန်
- Data distribution - ဒေတာဖြန့်ဝေမှုပုံစံများကို လေ့လာရန်
- Categorical data visualization - အမျိုးအစားခွဲဒေတာများကို နှိုင်းယှဉ်ဖော်ပြရန်

Seaborn ကို သင်ယူရန် အကောင်းဆုံးနည်းလမ်းမှာ လက်တွေ့လုပ်ဆောင်ကြည့်ရှုခြင်းဖြစ်ပါတယ်။ Official documentation တွင် ဥပမာများစွာရှိပါတယ်။

၂၀.၃။ Seaborn ဖြင့် Data Exploration ဥပမာ

Titanic ဒေတာစုကို အသုံးပြုပြီး ရှင်းပြချက်

အောက်ပါဥပမာမှာ Seaborn ကိုသုံးပြီး Titanic ဒေတာစုကို စူးစမ်းကြည့်ပါမယ်။

```
import seaborn as sns
import matplotlib.pyplot as plt

# Titanic ဒေတာစုကို ဖတ်ပါမယ်
titanic = sns.load_dataset('titanic')

# ဒေတာအချက်အလက်များကို အရင်ဆုံး ကြည့်ပါမယ်
print(titanic.head())

print("\nဒေတာအချက်အလက်များ:")
print(titanic.info())
```



Output:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	

	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False
1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True

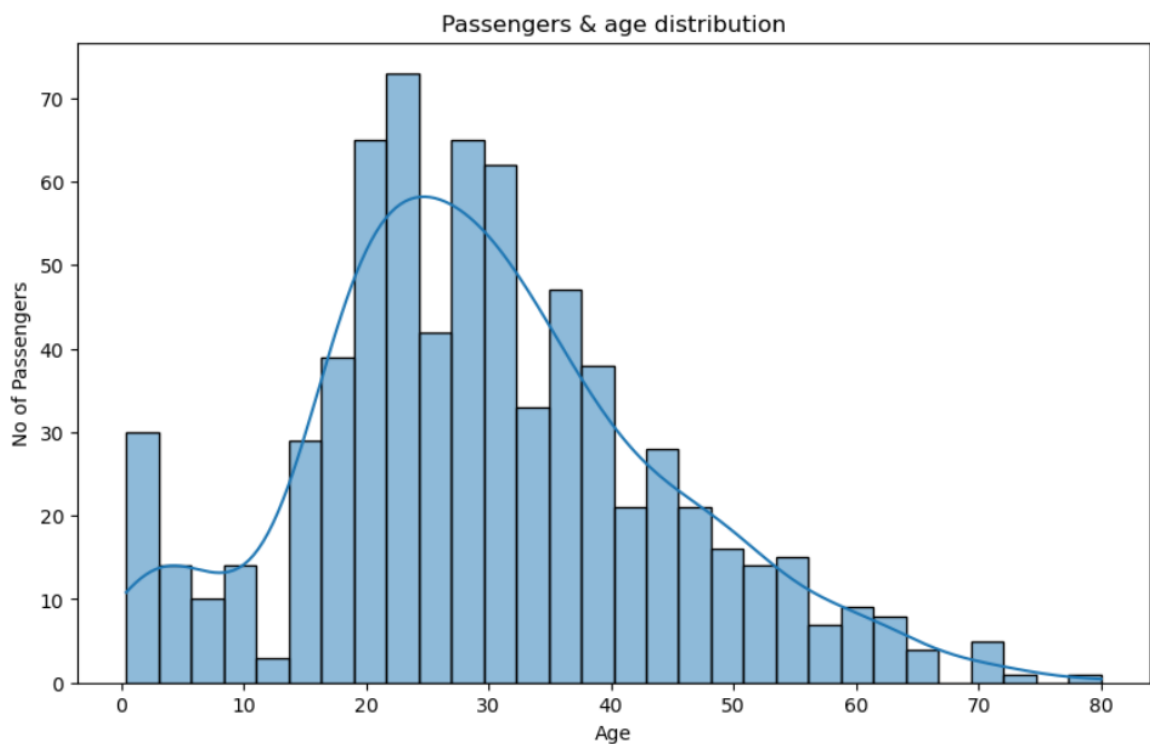
ဒေတာအချက်အလက်များ:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column          Non-Null Count  Dtype
---  -
0   survived        891 non-null    int64
1   pclass          891 non-null    int64
2   sex             891 non-null    object
3   age            714 non-null    float64
4   sibsp          891 non-null    int64
5   parch          891 non-null    int64
6   fare           891 non-null    float64
7   embarked       889 non-null    object
8   class          891 non-null    category
9   who            891 non-null    object
10  adult_male     891 non-null    bool
11  deck          203 non-null    category
12  embark_town    889 non-null    object
13  alive          891 non-null    object
14  alone          891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```

၂၀.၄။ အခြေခံ Statistical ပုံများ

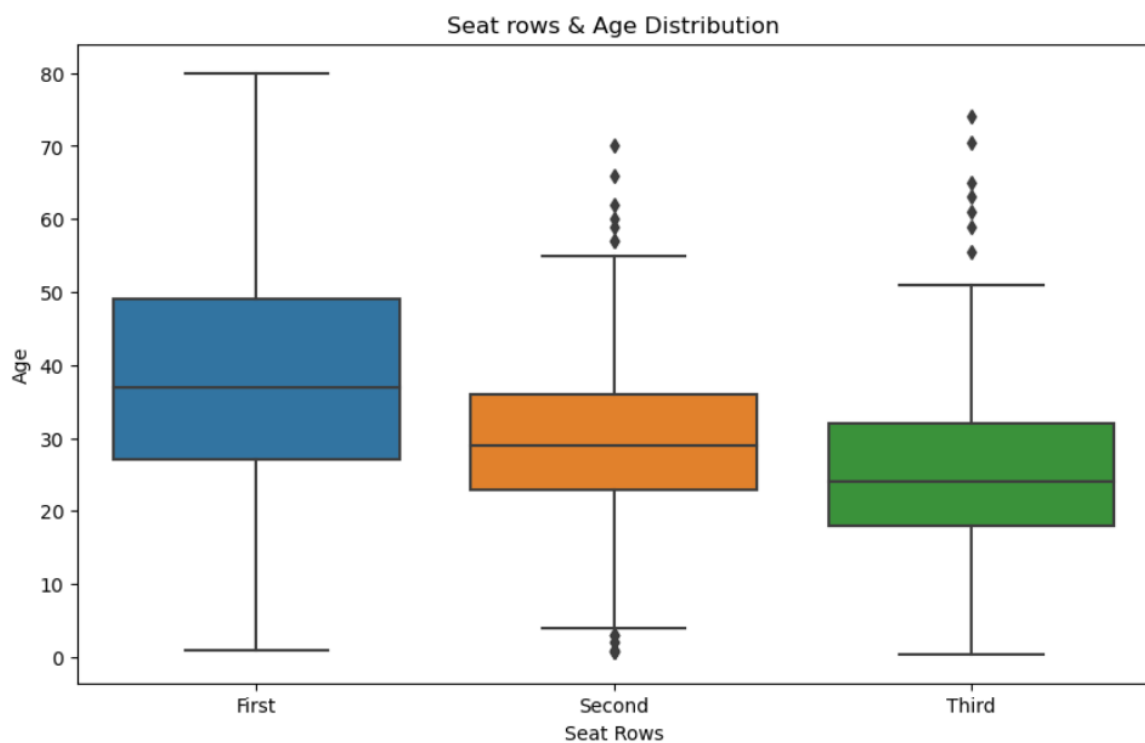
Histogram

```
plt.figure(figsize=(10,6))
sns.histplot(data=titanic, x='age', bins=30, kde=True)
plt.title('Passengers & Age Distributions')
plt.xlabel('Age')
plt.ylabel('Population')
plt.show()
```



Boxplot (အတန်းလိုက်နှိုင်းယှဉ်ခြင်း)

```
plt.figure(figsize=(10,6))
sns.boxplot(data=titanic, x='class', y='age')
plt.title('Seat Rows & Age Distribution' )
plt.xlabel('Seat Rows')
plt.ylabel('Age')
plt.show()
```



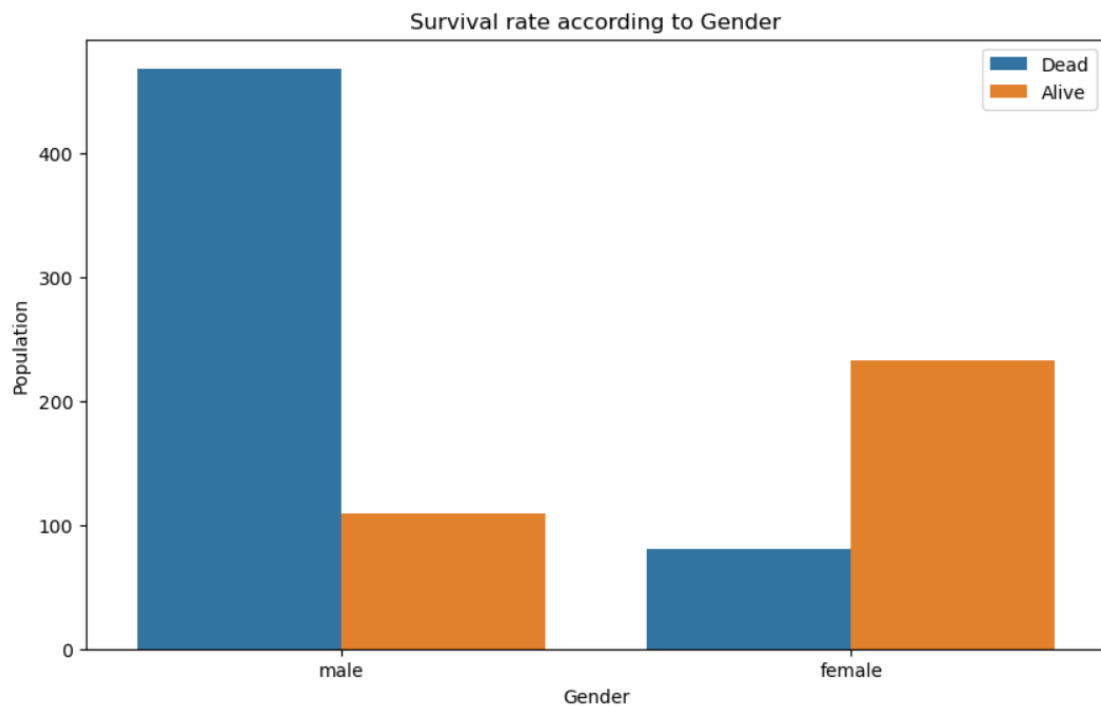
၂၀.၅။ Categorical Data များကို ဆန်းစစ်ခြင်း

Countplot (အမျိုးအစားခွဲခြင်း)

```
plt.figure(figsize=(10,6))
sns.countplot(data=titanic, x='sex', hue='survived')
plt.title('Survival Rate According to Gender')
plt.xlabel('Gender')
```

```
plt.ylabel('Population')  
plt.legend(['Dead', 'Alive'])  
plt.show()
```

Output:

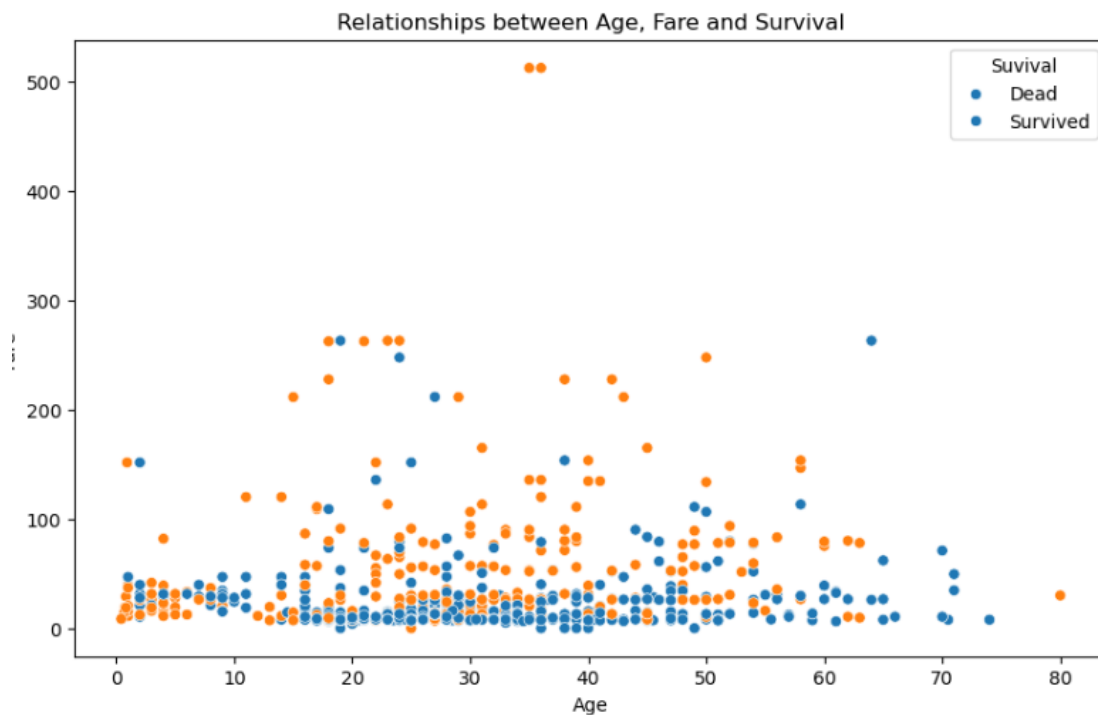


၂၀.၆။ Relationship များကို ဆန်းစစ်ခြင်း

Scatterplot (ဆက်စပ်မှုရှာဖွေခြင်း)

```
plt.figure(figsize=(10,6))  
sns.scatterplot(data=titanic, x='age', y='fare', hue='survived')  
plt.title( 'Relationships between Age, Fare and Survival' )  
plt.xlabel('Age')  
plt.ylabel('Fare')  
plt.legend(title='Survival Rate ', labels=['Dead', 'Survived'])  
plt.show()
```

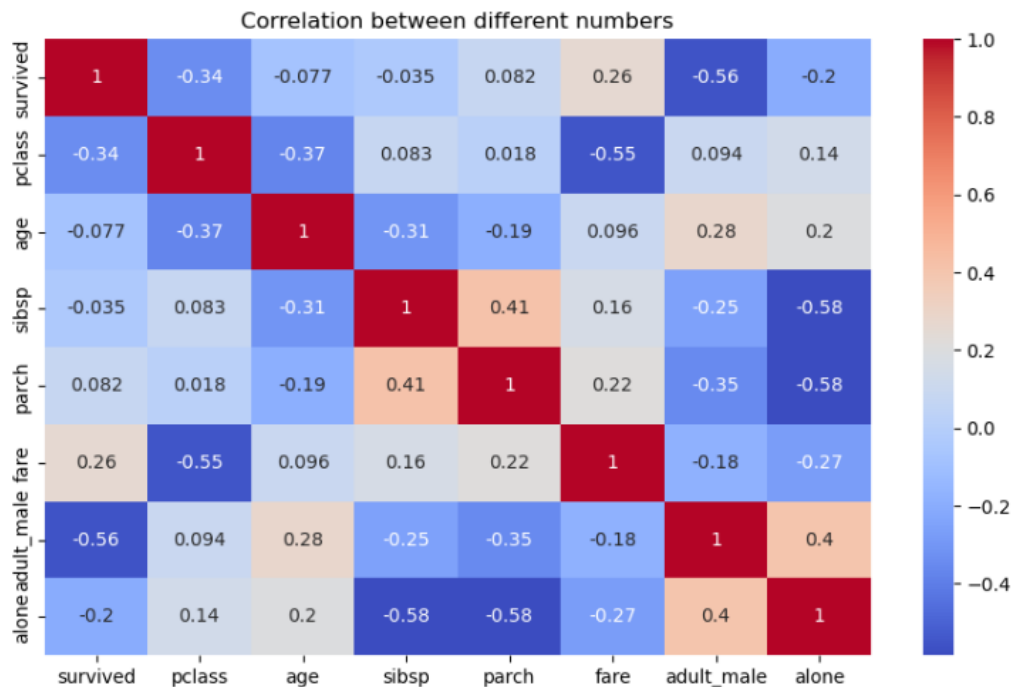
Output:



Heatmap (Correlation ဆန်းစစ်ခြင်း)

```
plt.figure(figsize=(10,6))
corr = titanic.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title(' Correlation between numbers ')
plt.show()
```

Output:

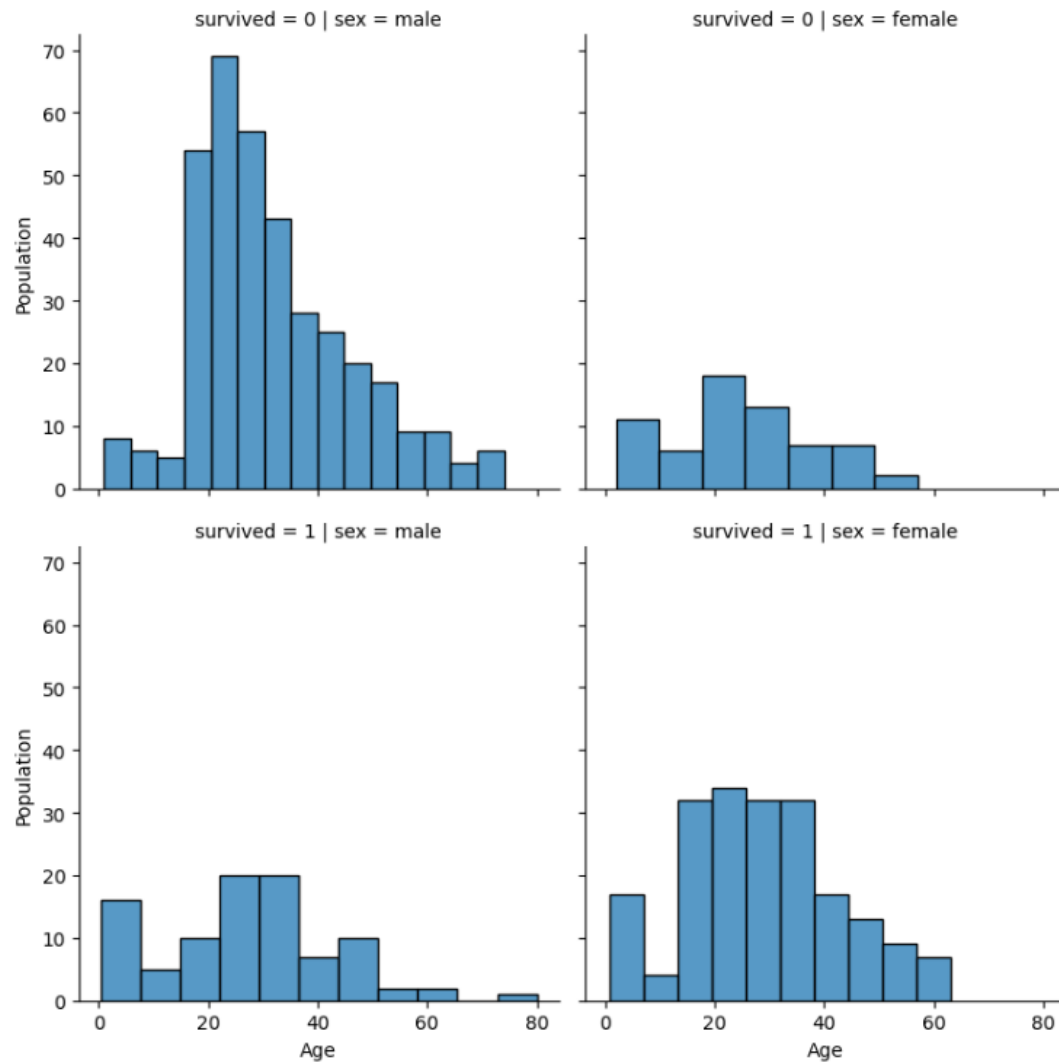


၂၀.၇။ Advanced Visualization

FacetGrid (အမျိုးမျိုးခွဲကြည့်ခြင်း)

```
g = sns.FacetGrid(titanic, col='sex', row='survived', height=4)
g.map(sns.histplot, 'age')
g.set_axis_labels('Age', 'Population')
plt.show()
```

Output:

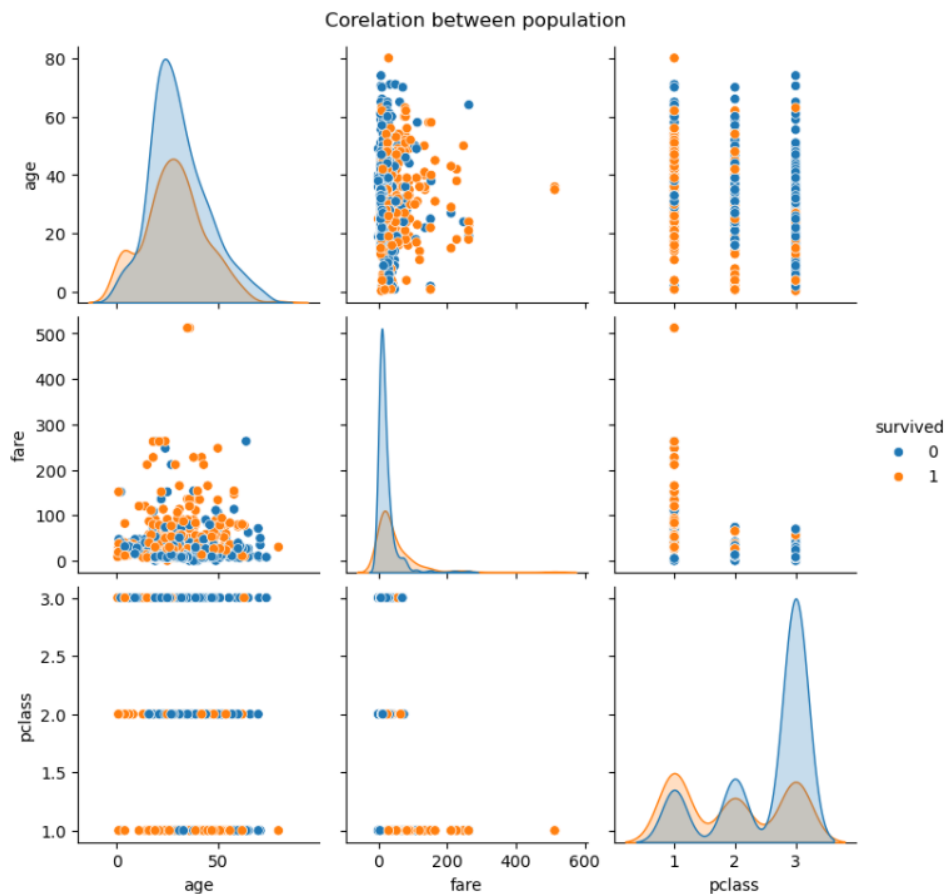


Pairplot (အားလုံးသော numeric variables များကို တစ်ပြိုင်နက်ကြည့်ခြင်း)

```
import seaborn as sns
import matplotlib.pyplot as plt

# Titanic ဒေတာစုကို ဖတ်ပါမယ်
titanic = sns.load_dataset('titanic')
sns.pairplot(titanic[['age', 'fare', 'pclass', 'survived']], hue='survived')
plt.suptitle('Corelation between population', y=1.02)
plt.show()
```

Output:



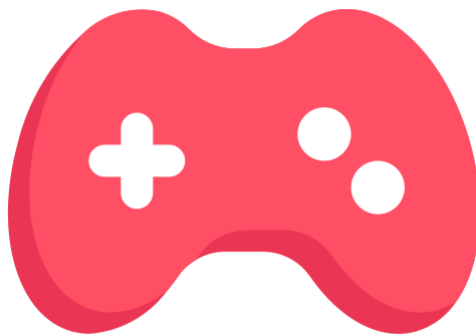
▣ ရှင်းလင်းချက်

1. **Histogram** - ဒေတာဖြန့်ဖြူးပုံကို နားလည်ရန် (ဥပမာ - အများဆုံးခရီးသည်များသည် အသက်ဘယ်အပိုင်းအခြားမှာရှိသလဲ)
2. **Boxplot** - အုပ်စုလိုက် နှိုင်းယှဉ်ကြည့်ရန် (ဥပမာ - ဘယ်တန်းစားခရီးသည်များက ပျမ်းမျှအသက်ကြီးသလဲ)
3. **Countplot** - အမျိုးအစားခွဲခြားနှင့် အရေအတွက်ကို ကြည့်ရန် (ဥပမာ - အမျိုးသမီးများက အမျိုးသားများထက် အသက်ပိုရှင်နိုင်ခြေရှိသလား)
4. **Heatmap** - variables များအကြား ဆက်စပ်မှုကို နားလည်ရန်

Seaborn သည် ဒေတာများကို အလွယ်တကူနှင့် အလှပဆုံး ပုံဖော်နိုင်သော library တစ်ခုဖြစ်ပါတယ်။ ဒေတာသိပ္ပံပညာရှင်များအတွက် အထူးအသုံးဝင်ပါတယ်။

အခန်း(၂၁)

Mini Games



၂၁.၁။ Tic Tac Toe Game

Game တွေကို ရေးသားခြင်းက Programming ကို စတင်လေ့လာသူတွေ အတွက် **အကောင်းဆုံး အလေ့အကျင့်** တစ်ခုဖြစ်ပါတယ်။ Game တွေကို ရေးသားခြင်းရဲ့ ကောင်းကျိုးတွေကို အောက်ပါအတိုင်း အဆင့်ဆင့် ရှင်းပြပါမယ်။

၁။ စိတ်ဝင်တစား အာရုံစူးစိုက်ပြီး လေ့လာနိုင်ခြင်း

- Game Development က ပျင်းစရာကောင်းတဲ့ Programming Concepts တွေကို **ပျော်စရာတွေ** နဲ့ လေ့လာခွင့်ပေးပါတယ်။
- ကိုယ်တိုင် ဖန်တီးထားတဲ့ Game ကို ကစားရတာကြောင့် **စိတ်အားထက်သန်မှု** ပိုရှိစေပါတယ်။

၂။ Programming Logic ကို လွယ်ကူစွာ နားလည်စေခြင်း

- Game တွေမှာ **Loop, Condition, Function, OOP** စတဲ့ အခြေခံ Programming Concepts တွေကို မဖြစ်မနေ အသုံးပြုရပါတယ်။
- ဥပမာ:
 - Player Movement → Variable & Condition
 - Score System → Data Structure & Algorithm
 - Enemy AI → Logic & Problem Solving

၃။ Creativity နဲ့ Problem-Solving Skill တိုးတက်စေခြင်း

- Game တစ်ခုဖန်တီးဖို့ **ဖန်တီးနိုင်စွမ်း (Creativity)** နဲ့ **ပြဿနာဖြေရှင်းနိုင်စွမ်း (Problem-Solving)** တို့ကို လေ့ကျင့်ပေးတယ်။
- Bug တွေကို Debug လုပ်ရင်း **Logical Thinking** တိုးတက်လာမယ်။



၄။ Project-Based Learning (လက်တွေ့လုပ်ဆောင်ခြင်း)

- Game Development က **သေးငယ်တဲ့ Project** ကနေ **ကြီးမားတဲ့ Project** အထိ တဖြည်းဖြည်း တည်ဆောက်နိုင်တယ်။
- ဥပမာ:
 - **Tic Tac Toe** → အစပိုင်း Logic လေ့ကျင့်ရန် ရည်ရွယ်ပါတယ်။
 - **Snake Game** → Physics & Animation လေ့လာရန် ရည်ရွယ်ပါတယ်။

၅။ အခြား Fields အတွက် အခြေခံကောင်းဖြစ်စေခြင်း

- Game Dev မှာ သင်ယူထားတဲ့ **Coding Skills, Algorithm, Optimization** တွေကို Web Dev, Mobile App, AI စတဲ့ နယ်ပယ်တွေမှာပါ အသုံးပြုနိုင်တယ်။

အကျဉ်းချုပ် ပြောရမယ်ဆိုရင် Game Development က Programming ကို ပျော်ပျော်ရွှင်ရွှင် လေ့လာနိုင်ပြီး Problem-Solving, Creativity, Logic Building စွမ်းရည်တွေ တိုးတက်စေတဲ့အတွက် စတင်လေ့လာသူတိုင်း လုပ်သင့်တဲ့ အလေ့အကျင့် ကောင်းတစ်ခုဖြစ်ပါတယ်။

Output:

```
Tic Tac Toe Game
Player 1: X , Player 2: 0
-----
0 | 0 | 0
-----
0 | 0 | 0
-----
0 | 0 | 0
-----
Player 1 turn...
Enter Input:[1-9]:|
```

```
Tic Tac Toe Game
Player 1: X , Player 2: 0
-----
X | 0 | 0
-----
0 | 0 | 0
-----
0 | 0 | 0
-----
Player 2 turn...
Enter Input:[1-9]:
```

Source Code:

```
b=['0','0','0','0','0','0','0','0','0']
def print_board():
    print(' Tic Tac Toe Game ')
    print('Player 1: X , Player 2: O ')
    print('-----')
    print(f' {b[0]} | {b[1]} | {b[2]}')
    print('-----')
    print(f' {b[3]} | {b[4]} | {b[5]}')
    print('-----')
    print(f' {b[6]} | {b[7]} | {b[8]}')
    print('-----')

def get_input():
    return int(input('Enter Input:[1-9]:'))

def play(player):
    res=1
    if player==1:
        symbol='X'
    else:
        symbol='O'

    print(f'Player {player} turn...')
    input=get_input()
    if input==1 and b[0]=='0':
        b[0]=symbol
    elif input==2 and b[1]=='0':
        b[1]=symbol
    elif input==3 and b[2]=='0':
        b[2]=symbol
    elif input==4 and b[3]=='0':
        b[3]=symbol
```

```
elif input==5 and b[4]=='0':
    b[4]=symbol
elif input==6 and b[5]=='0':
    b[5]=symbol
elif input==7 and b[6]=='0':
    b[6]=symbol
elif input==8 and b[7]=='0':
    b[7]=symbol
elif input==9 and b[8]=='0':
    b[8]=symbol
else:
    res=0

return res
```

```
def check_winner(player):
    res=0
    if player==1:
        if (b[0]=='X' and b[1]=='X' and b[2]=='X' or
            b[3]=='X' and b[4]=='X' and b[5]=='X' or
            b[6] == 'X' and b[7] == 'X' and b[8] == 'X' or

            b[0] == 'X' and b[3] == 'X' and b[6] == 'X' or
            b[1] == 'X' and b[4] == 'X' and b[7] == 'X' or
            b[2] == 'X' and b[5] == 'X' and b[8] == 'X' or

            b[0] == 'X' and b[4] == 'X' and b[8] == 'X' or
            b[2] == 'X' and b[4] == 'X' and b[6] == 'X'):
            res=1
    if player==2:
        if (b[0]=='O' and b[1]=='O' and b[2]=='O' or
            b[3]=='O' and b[4]=='O' and b[5]=='O' or
            b[6] == 'O' and b[7] == 'O' and b[8] == 'O' or

            b[0] == 'O' and b[3] == 'O' and b[6] == 'O' or
            b[1] == 'O' and b[4] == 'O' and b[7] == 'O' or
            b[2] == 'O' and b[5] == 'O' and b[8] == 'O' or
```



```
b[0] == 'O' and b[4] == 'O' and b[8] == 'O' or  
b[2] == 'O' and b[4] == 'O' and b[6] == 'O'):  
    res=2  
return res
```

```
def winning_message(player):  
    print('Congratulations....')  
    print('-----')  
    print(f'player {player} won...')  
    print('-----')  
    print('Game Over')
```

```
def check_draw():  
    res=0  
    if (b[0]!='O' and  
        b[1]!='O' and  
        b[2] != 'O' and  
  
        b[3] != 'O' and  
        b[4] != 'O' and  
        b[5] != 'O' and  
  
        b[6] != 'O' and  
        b[7] != 'O' and  
        b[8] != 'O'):  
        res=1  
    else:  
        res=0  
    return res
```

```
def draw_message():  
    print('Opps No more moves.....')  
    print('-----')  
    print('It is a Draw..')  
    print('-----')  
    print('Game Over')
```



```
def main():
    count=0
    player=0
    game_over=0
    while not game_over:
        print_board()

        if count%2==0:
            player=1
        else:
            player=2

        if play(player)==0:
            print('invalid inputs...')
            count-=1
        if check_winner(player)==1 or check_winner(player)==2:
            print_board()
            winning_message(player)
            game_over=1

        if check_draw():
            print_board()
            draw_message()
            game_over=1

        count+=1

if __name__=='__main__':
    main()
```

ရှင်းလင်းချက်။

Game Logic:

1. 3x3 grid (game board) ကွက်အလွတ်နဲ့စပါမယ်
2. Player တစ်ယောက်က 'X' ၊ နောက်တစ်ယောက်က 'O' နဲ့ကစားရပါမယ်
3. အလှည့်ကျရွေးခွယ်ပြီး input (1-9) ကိုရေးသွင်းကြရပါမယ်
4. မည်သူ့မဆို ၃ လုံးတန်းရင် အနိုင်ရပါမယ်
5. Grid ကွက်(board) ပြည့်သွားရင် သရေကျပါမယ်

အသေးစိတ်ရှင်းလင်းချက်။

1. print_board()
 - ဂရစ်ကွက်ကိုပြသပေးတဲ့ function ဖြစ်ပါတယ်။

Code:

```
def print_board():
```

```
    print(' Tic Tac Toe Game ')\n    print('Player 1: X , Player 2: O ')\n    print('-----')\n    print(f' {b[0]} | {b[1]} | {b[2]}')\n    print('-----')\n    print(f' {b[3]} | {b[4]} | {b[5]}')\n    print('-----')\n    print(f' {b[6]} | {b[7]} | {b[8]}')\n    print('-----')
```

2. check_winner()
 - အနိုင်ရှိမရှိစစ်ဆေးတဲ့ function ဖြစ်ပါတယ်။
 - player 1 နဲ့ player 2 အတွက် winning conditions တွေကို ရေးသားရပါမယ်။

Code:

```
def check_winner(player):
    res=0
    if player==1:
        if (b[0]=='X' and b[1]=='X' and b[2]=='X' or
            b[3]=='X' and b[4]=='X' and b[5]=='X' or
            b[6] == 'X' and b[7] == 'X' and b[8] == 'X' or

            b[0] == 'X' and b[3] == 'X' and b[6] == 'X' or
            b[1] == 'X' and b[4] == 'X' and b[7] == 'X' or
            b[2] == 'X' and b[5] == 'X' and b[8] == 'X' or

            b[0] == 'X' and b[4] == 'X' and b[8] == 'X' or
            b[2] == 'X' and b[4] == 'X' and b[6] == 'X'):
            res=1
    if player==2:
        if (b[0]=='O' and b[1]=='O' and b[2]=='O' or
            b[3]=='O' and b[4]=='O' and b[5]=='O' or
            b[6] == 'O' and b[7] == 'O' and b[8] == 'O' or

            b[0] == 'O' and b[3] == 'O' and b[6] == 'O' or
            b[1] == 'O' and b[4] == 'O' and b[7] == 'O' or
            b[2] == 'O' and b[5] == 'O' and b[8] == 'O' or

            b[0] == 'O' and b[4] == 'O' and b[8] == 'O' or
            b[2] == 'O' and b[4] == 'O' and b[6] == 'O'):
            res=2
    return res
```

3. check_draw()

- Grid Board (9) ကွက်ပြည့်မပြည့်စစ်ဆေးတဲ့ function ဖြစ်ပါတယ်။
- Grid Board 9 ကွက်စလုံး ဖြည့်ပြီးသွားပြီး အနိုင်အရှုံး မပေါ်ပေါက်ခဲ့ဘူးဆိုရင် Game ကို ရပ်ပါမယ်။

Code:

```
def check_draw():
    res=0
    if (b[0]!='0' and
        b[1]!='0' and
        b[2] != '0' and

        b[3] != '0' and
        b[4] != '0' and
        b[5] != '0' and

        b[6] != '0' and
        b[7] != '0' and
        b[8] != '0'):
        res=1
    else:
        res=0
    return res
```

4. winning_message()

– check_winner() function ကနေ player 1 သို့မဟုတ် player 2 အနိုင်ရသွားခဲ့မယ်ဆိုရင် winning message ထုတ်ပြပေးပါမယ်။

Code:

```
def winning_message(player):
    print('Congratulations....')
    print('-----')
    print(f'player {player} won...')
    print('-----')
    print('Game Over')
```

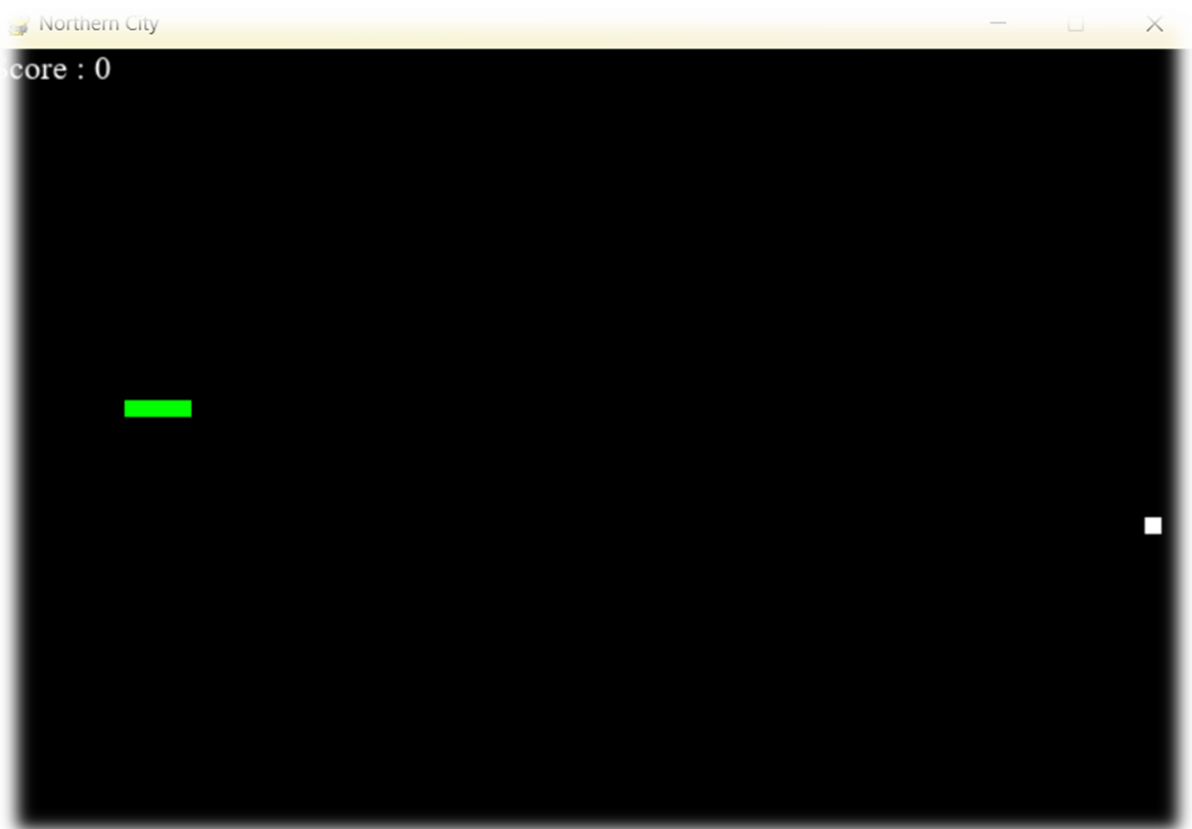
5. draw_message()

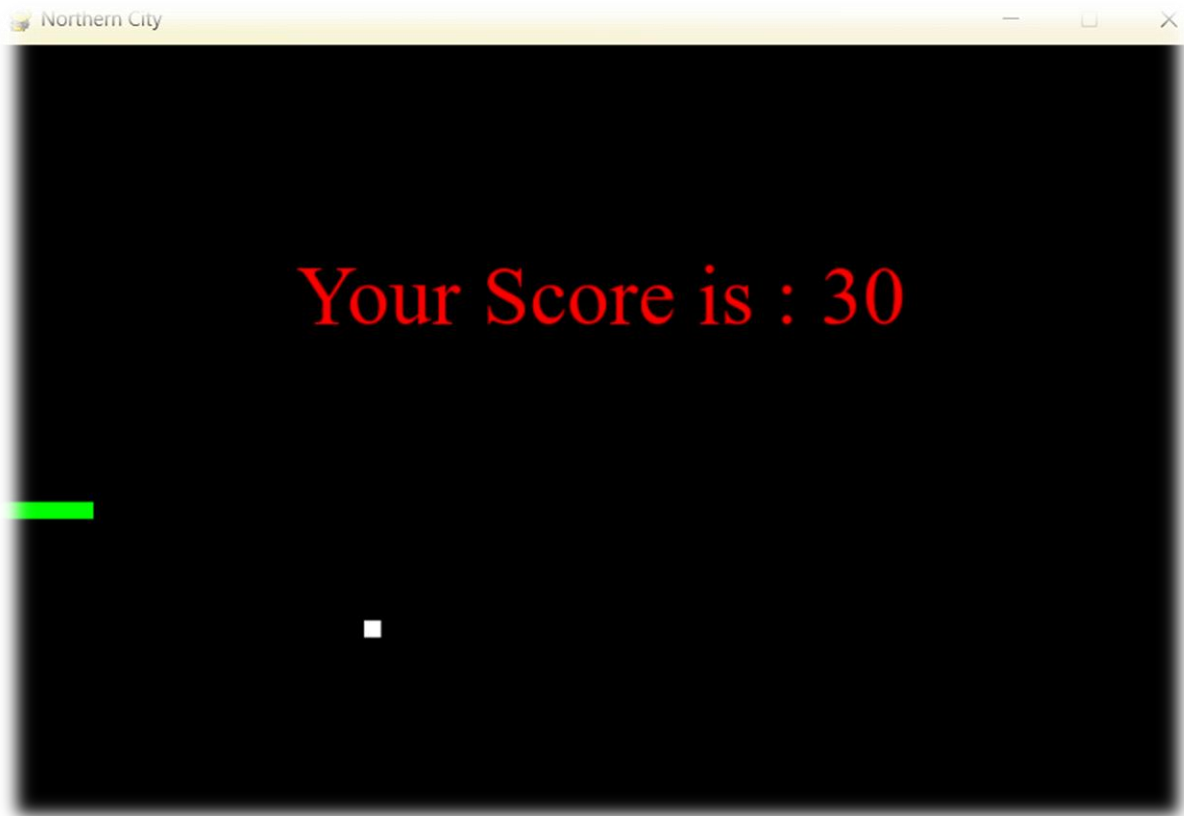
– check_draw() function ကနေ return true ဖြစ်ခဲ့မယ်ဆိုရင် draw message ကို ထုတ်ပြရမှာ ဖြစ်ပါတယ်။

```
def draw_message():  
    print('Opps No more moves.....')  
    print('-----')  
    print('It is a Draw..')  
    print('-----')  
    print('Game Over')
```

၂၁.၂။ Snake Game

Output:





Source Code:

```
# importing libraries
import pygame
import time
import random

def main():
    #speed
    snake_speed = 15

    # board size
    width = 720
    height = 480
```



```
# basic colors
black = pygame.Color(0, 0, 0)
white = pygame.Color(255, 255, 255)
red = pygame.Color(255, 0, 0)
green = pygame.Color(0, 255, 0)
blue = pygame.Color(0, 0, 255)

# init pygame
pygame.init()

# init game window
pygame.display.set_caption('Northern City')
game_window = pygame.display.set_mode((width, height))

# FPS (frames per second) controller
fps = pygame.time.Clock()

# defining snake default position
snake_position = [100, 50]

# defining first 4 blocks of snake body
snake_body = [[100, 50],[90, 50],[80, 50],[70, 50]]
# fruit position
food_position = [random.randrange(1, (width // 10)) * 10, random.randrange(1,
(height // 10)) * 10]

food_spawn = True

# setting default snake direction towards
# right
direction = 'RIGHT'
change_to = direction

# initial score
score = 0
```




```
# displaying Score function
def display_result(choice, color, font, size):

    # creating font object score_font
    score_font = pygame.font.SysFont(font, size)

    # create the display surface object
    # score_area
    score_area = score_font.render('Score : ' + str(score), True, color)

    # create a rectangular object for the text
    # surface object
    score_rect = score_area.get_rect()

    # displaying text
    game_window.blit(score_area, score_rect)

# game over function
def game_over():

    # object font
    my_font = pygame.font.SysFont('times new roman', 50)

    # creating a text surface on which text
    # will be drawn
    game_over_surface = my_font.render('Your Score is : ' + str(score), True, red)

    # create a rectangular object for the text
    # surface object
    game_over_rect = game_over_surface.get_rect()

    # setting position of the text
    game_over_rect.midtop = (width / 2, height / 4)

    # blit will draw the text on screen
```



```
game_window.blit(game_over_surface, game_over_rect)
pygame.display.flip()

# after 2 seconds we will quit the program
time.sleep(2)

# deactivating pygame library
pygame.quit()

# quit the program
quit()

# game inputs
while True:

    # handling key events
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                direction = 'UP'
            if event.key == pygame.K_DOWN:
                direction = 'DOWN'
            if event.key == pygame.K_LEFT:
                direction = 'LEFT'
            if event.key == pygame.K_RIGHT:
                direction = 'RIGHT'

    # Moving the snake
    if direction == 'UP':
        snake_position[1] -= 10
    if direction == 'DOWN':
        snake_position[1] += 10
    if direction == 'LEFT':
        snake_position[0] -= 10
```



```

if direction == 'RIGHT':
    snake_position[0] += 10

# game logics
# Snake body growing codes
snake_body.insert(0, list(snake_position))

# if food and snakes collide then increase scores
# score will be incremented by 10
if snake_position[0] == food_position[0] and snake_position[1] ==
food_position[1]:
    score += 10
    food_position = [random.randrange(1, (width // 10)) * 10, random.randrange(1,
(height // 10)) * 10]
else:
    snake_body.pop()

food_spawn = True
game_window.fill(black)

for pos in snake_body:
    pygame.draw.rect(game_window, green, pygame.Rect(pos[0], pos[1], 10, 10))
    pygame.draw.rect(game_window, white, pygame.Rect(food_position[0],
food_position[1], 10, 10))

# Check if the snake hit the wall
if snake_position[0] < 0 or snake_position[0] > width - 10:
    game_over()
if snake_position[1] < 0 or snake_position[1] > height - 10:
    game_over()

# displaying score continuously
display_result(1, white, 'times new roman', 20)

```



```
# Refresh game screen
pygame.display.update()

# Frame Per Second /Refresh Rate
fps.tick(snake_speed)

if __name__ == '__main__':
    main()
```

၂၁.၂.၁။ အသေးစိတ်ရှင်းလင်းချက်။

Python နဲ့ Snake Game ရေးသားဖို့အတွက် pygame library ကိုအသုံးပြုပါမယ်။ ဒီ game မှာ အခြေခံ Game Loop, Keyboard Controls, Collision Detection နဲ့ Score System တို့ကို လေ့လာရမှာဖြစ်ပါတယ်။

၁။ လိုအပ်သော Libraries များ

```
import pygame
import time
import random
```

- **pygame** → Game Development အတွက် အဓိက Library
- **time** → Game Speed ထိန်းချုပ်ဖို့

- `random` → Snake Food ကို Random ထုတ်ဖို့
-

၂။ Game Initialization

```
snake_speed = 15

# board size
width = 720
height = 480

# basic colors
black = pygame.Color(0, 0, 0)
white = pygame.Color(255, 255, 255)
red = pygame.Color(255, 0, 0)
green = pygame.Color(0, 255, 0)
blue = pygame.Color(0, 0, 255)

# init pygame
pygame.init()

# init game window
pygame.display.set_caption('Northern City')
game_window = pygame.display.set_mode((width, height))

# FPS (frames per second) controller
fps = pygame.time.Clock()

# defining snake default position
snake_position = [100, 50]

# defining first 4 blocks of snake body
```

```
snake_body = [[100, 50],[90, 50],[80, 50],[70, 50]]
# fruit position
food_position = [random.randrange(1, (width // 10)) * 10, random.randrange(1,
(height // 10)) * 10]

food_spawn = True

# setting default snake direction towards
# right
direction = 'RIGHT'
change_to = direction

# initial score
score = 0
```

- `pygame.init()` → Pygame ကို စတင်သုံးဖို့ ရေးရပါမယ်။
 - `set_mode()` → Game Window Size သတ်မှတ်ခြင်း ဖြစ်ပါတယ်။
 - RGB Colors → Snake, Food, Background အတွက် အရောင်သတ်မှတ်ခြင်း
-

၃။ Snake & Food ဖန်တီးခြင်း

Snake ကိုဖန်တီးပြီး Movement ထည့်ခြင်း

```
#create snake body
for pos in snake_body:
    pygame.draw.rect(game_window, green, pygame.Rect(pos[0], pos[1], 10, 10))

#create food
pygame.draw.rect(game_window, white, pygame.Rect(food_position[0],
food_position[1], 10, 10))
```

Food ကို Random ထုတ်ခြင်း

```
food_x = round(random.randrange(1, width - block_size) / block_size) * block_size
food_y = round(random.randrange(1, height - block_size) / block_size) * block_size
```

```
food_position = [random.randrange(1, (width // 10)) * 10, random.randrange(1,
(height // 10)) * 10]
```

၄။ Main Game Loop

while True:

```
# handling key events
for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_UP:
            direction = 'UP'
        if event.key == pygame.K_DOWN:
            direction = 'DOWN'
        if event.key == pygame.K_LEFT:
            direction = 'LEFT'
        if event.key == pygame.K_RIGHT:
            direction = 'RIGHT'
```



```
# Moving the snake
if direction == 'UP':
    snake_position[1] -= 10
if direction == 'DOWN':
    snake_position[1] += 10
if direction == 'LEFT':
    snake_position[0] -= 10
if direction == 'RIGHT':
    snake_position[0] += 10

# game logics
# Snake body growing codes
snake_body.insert(0, list(snake_position))

# if food and snakes collide then increase scores
# score will be incremented by 10
if snake_position[0] == food_position[0] and snake_position[1] ==
food_position[1]:
    score += 10
    food_position = [random.randrange(1, (width // 10)) * 10, random.randrange(1,
(height // 10)) * 10]
else:
    snake_body.pop()

food_spawn = True
game_window.fill(black)

for pos in snake_body:
    pygame.draw.rect(game_window, green, pygame.Rect(pos[0], pos[1], 10, 10))
    pygame.draw.rect(game_window, white, pygame.Rect(food_position[0],
food_position[1], 10, 10))
```




```

# Check if the snake hit the wall
if snake_position[0] < 0 or snake_position[0] > width - 10:
    game_over()
if snake_position[1] < 0 or snake_position[1] > height - 10:
    game_over()

# displaying score continuously
display_result(1, white, 'times new roman', 20)

# Refresh game screen
pygame.display.update()

# Frame Per Second /Refresh Rate
fps.tick(snake_speed)

```

၅။ Game Mechanics ရှင်းလင်းချက်

1. Snake Movement

- **Keyboard Arrow Keys** နဲ့ ထိန်းချုပ်မယ် (LEFT, RIGHT, UP, DOWN)
- direction ကို update လုပ်ပြီး မြွေကိုရွေ့စေမယ်။

2. Food စားခြင်း

- Snake က Food ကိုထိရင် **Snake Length တိုးမယ်**
- **New Food** ကို Random နေရာမှာ ထုတ်ပေးမယ်။



3. Game Over Conditions

- နံရံ (Wall) နဲ့ ထိရင် Game Over

4. Score System

- snake_length ကို Score အဖြစ်သတ်မှတ်နိုင်တယ်။

Python နဲ့ Snake Game ရေးသားခြင်းက Programming အခြေခံ (Loop, Condition, Function, OOP) တွေကို လက်တွေ့လေ့ကျင့်ဖို့ အကောင်းဆုံးနည်းလမ်းဖြစ်ပါတယ်။



အခန်း(၂၂)

Python & Database



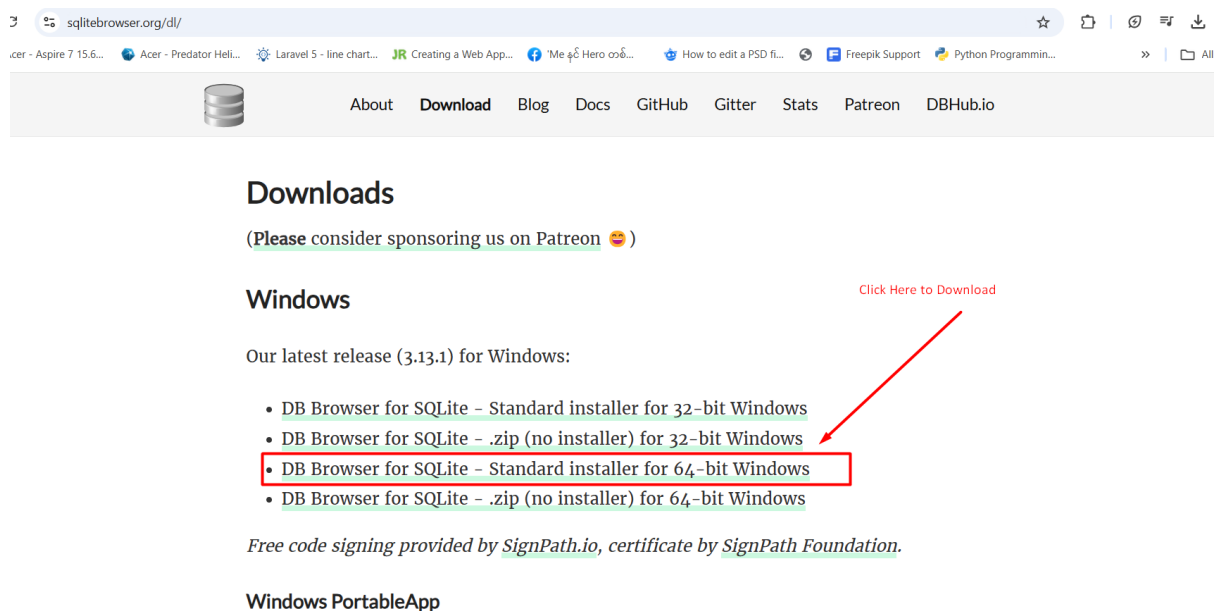


❑ Python and Database

Python နဲ့ SQLite3 ကို သုံးပြီး Product CRUD (Create, Read, Update, Delete) operation တွေကို ရေးကြည့်ရအောင်။ Project structure ကို အောက်ပါအတိုင်း ဖန်တီးပါမယ်။

Sqlite Browser ကို ဒေါင်းလုပ်ဆွဲပြီး database file တစ်ခုဆောက်ရပါမယ်။

<https://sqlitebrowser.org/dl/>



sqlitebrowser.org/dl/

About Download Blog Docs GitHub Gitter Stats Patreon DBHub.io

Downloads

(Please consider sponsoring us on Patreon 🙏)

Windows

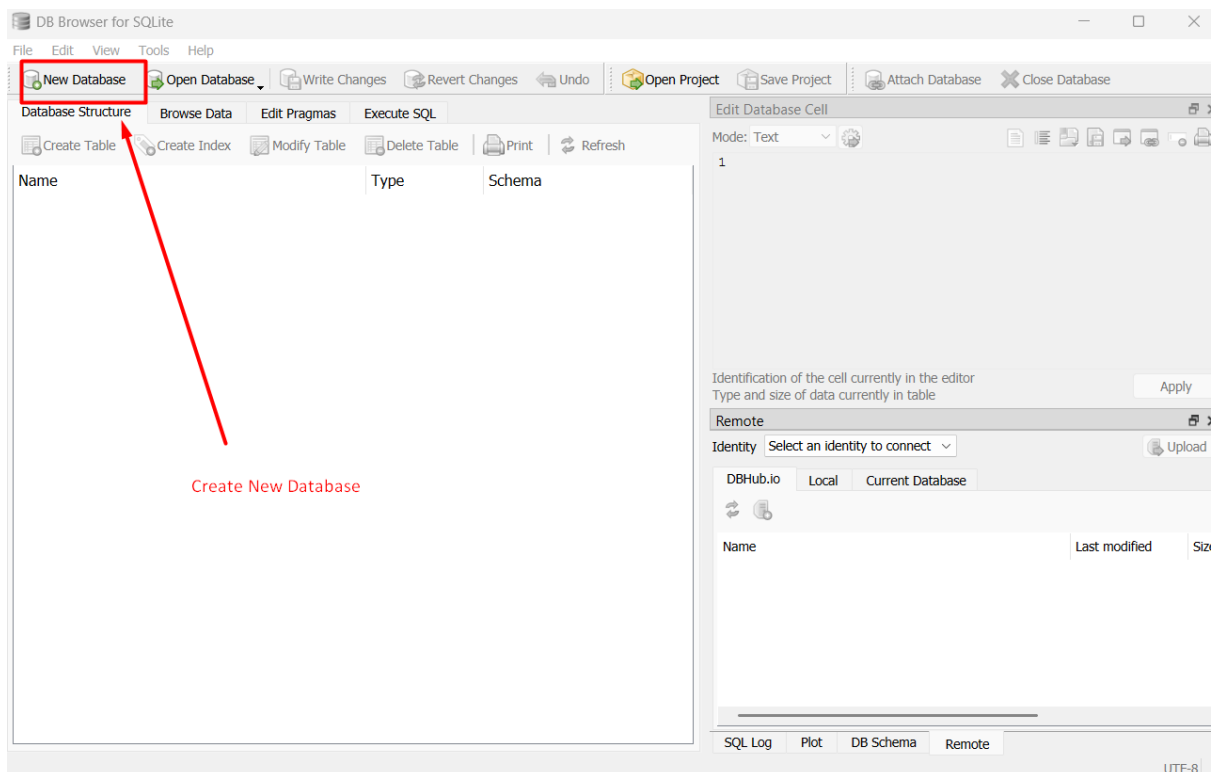
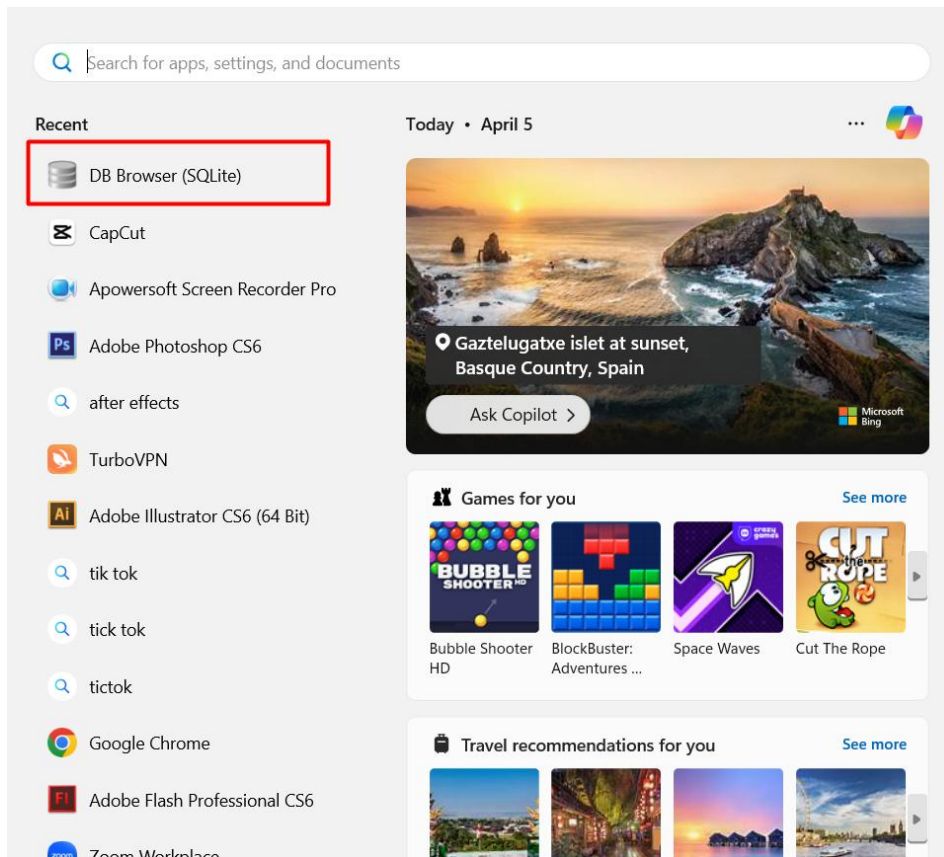
Our latest release (3.13.1) for Windows:

- DB Browser for SQLite - Standard installer for 32-bit Windows
- DB Browser for SQLite - .zip (no installer) for 32-bit Windows
- DB Browser for SQLite - Standard installer for 64-bit Windows
- DB Browser for SQLite - .zip (no installer) for 64-bit Windows

Free code signing provided by SignPath.io, certificate by SignPath Foundation.

Windows PortableApp

၂၂.၁။ Download ဆွဲပြီး install လုပ်ပါ။ ပြီးရင် database ဆောက်ရပါမယ်။





Edit table definition

Table
products

▼ Advanced

Fields Index Constraints Foreign Keys Check Constraints

Add Remove Move to top Move up Move down Move to bottom

Name	Type	NN	PK	AI	U	Default	Check	Collation
id	INTEGER	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
name	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
price	INTEGER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
quantity	INTEGER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			

```
1 CREATE TABLE "products" (  
2     "id"    INTEGER,  
3     "name"  TEXT,  
4     "price" INTEGER,  
5     "quantity" INTEGER,  
6     PRIMARY KEY ("id" AUTOINCREMENT)  
7 );
```

OK Cancel



၂.၂။ Project Structure

```
product_crud_app/  
├── models/  
│   └── product.py      # Product class အတွက် file  
├── services/  
│   └── product_service.py # CRUD operations အတွက် service file  
├── main.py             # Main application file  
└── database.db         # SQLite database file
```

Step 1: Product Class File

models/product.py မှာ Product class ကို ဖန်တီးပါမယ်။

```
# models/product.py
```

```
class Product:  
    def __init__(self, id=None, name=None, price=None, quantity=None):  
        self.id = id  
        self.name = name  
        self.price = price  
        self.quantity = quantity  
  
    def __str__(self):  
        return f"ID: {self.id}, Name: {self.name}, Price: {self.price}, Quantity: {self.quantity}"
```

Step 2: CRUD Service File

services/product_service.py မှာ SQLite3 နဲ့ CRUD operations တွေကို ရေးပါမယ်။



Python Code:

```
# services/product_service.py

import sqlite3
from models.product import Product

# Database connection
def get_db_connection():
    conn = sqlite3.connect('database.db')
    conn.row_factory = sqlite3.Row
    return conn

# Create table if not exists
def create_table():
    conn = get_db_connection()
    conn.execute("""
        CREATE TABLE IF NOT EXISTS products (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            price REAL NOT NULL,
            quantity INTEGER NOT NULL
        )
    """)
    conn.commit()
    conn.close()

# Add a new product
def add_product(product):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO products (name, price, quantity)
        VALUES (?, ?, ?)
    """, (product.name, product.price, product.quantity))
    conn.commit()
    conn.close()

# Get all products
def get_all_products():
    conn = get_db_connection()
```



```
cursor = conn.cursor()
cursor.execute('SELECT * FROM products')
rows = cursor.fetchall()
conn.close()

products = []
for row in rows:
    product = Product(id=row['id'], name=row['name'], price=row['price'], quantity=row['quantity'])
    products.append(product)
return products

# Get a product by ID
def get_product_by_id(product_id):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM products WHERE id = ?', (product_id,))
    row = cursor.fetchone()
    conn.close()

    if row:
        return Product(id=row['id'], name=row['name'], price=row['price'], quantity=row['quantity'])
    return None

# Update a product
def update_product(product):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        UPDATE products
        SET name = ?, price = ?, quantity = ?
        WHERE id = ?
    """, (product.name, product.price, product.quantity, product.id))
    conn.commit()
    conn.close()

# Delete a product
def delete_product(product_id):
    conn = get_db_connection()
    cursor = conn.cursor()
```

```
cursor.execute('DELETE FROM products WHERE id = ?', (product_id,))
conn.commit()
conn.close()
```

Step 3: Main Application File

main.py မှာ user interaction နဲ့ CRUD operations တွေကို လုပ်ဆောင်ပါမယ်။

Python Code:

main.py

```
from services.product_service import *
from models.product import Product
```

```
def show_menu():
```

```
    print("\n--- Product CRUD Menu ---")
    print("1. Add Product")
    print("2. View All Products")
    print("3. View Product by ID")
    print("4. Update Product")
    print("5. Delete Product")
    print("6. Exit")
```

```
def add_product_ui():
```

```
    name = input("Enter product name: ")
    price = float(input("Enter product price: "))
    quantity = int(input("Enter product quantity: "))
    product = Product(name=name, price=price, quantity=quantity)
    add_product(product)
    print("Product added successfully!")
```

```
def view_all_products_ui():
```

```
    products = get_all_products()
    if not products:
        print("No products found!")
    else:
        for product in products:
            print(product)
```

```
def view_product_by_id_ui():
    product_id = int(input("Enter product ID: "))
    product = get_product_by_id(product_id)
    if product:
        print(product)
    else:
        print("Product not found!")

def update_product_ui():
    product_id = int(input("Enter product ID to update: "))
    product = get_product_by_id(product_id)
    if product:
        name = input(f"Enter new name ({product.name}): ") or product.name
        price = float(input(f"Enter new price ({product.price}): ") or product.price)
        quantity = int(input(f"Enter new quantity ({product.quantity}): ") or product.quantity)
        updated_product = Product(id=product_id, name=name, price=price, quantity=quantity)
        update_product(updated_product)
        print("Product updated successfully!")
    else:
        print("Product not found!")

def delete_product_ui():
    product_id = int(input("Enter product ID to delete: "))
    product = get_product_by_id(product_id)
    if product:
        delete_product(product_id)
        print("Product deleted successfully!")
    else:
        print("Product not found!")

def main():
    create_table() # Create table if not exists

    while True:
        show_menu()
        choice = input("\nEnter your choice (1-6): ")

        if choice == "1":
            add_product_ui()
```

```
elif choice == "2":
    view_all_products_ui()
elif choice == "3":
    view_product_by_id_ui()
elif choice == "4":
    update_product_ui()
elif choice == "5":
    delete_product_ui()
elif choice == "6":
    print("Exiting the program. Goodbye!")
    break
else:
    print("Invalid choice! Please choose a number between 1 and 6.")

if __name__ == "__main__":
    main()
```

Step 4: Testing the Program

1. Run the Program:

Terminal မှာ main.py ကို run ပါ။

```
python main.py
```

2. Add Products:

- Menu ကနေ option 1 ကို ရွေးပြီး product တွေ ထည့်ပါ။

- Example:

Enter product name: Laptop

Enter product price: 1200.50

Enter product quantity: 10

3. View All Products:

- Menu ကနေ option 2 ကို ရွေးပြီး product တွေကို ကြည့်ပါ။

4. Update a Product:

- Menu ကနေ option 4 ကို ရွေးပြီး product တစ်ခုကို update လုပ်ပါ။

5. Delete a Product:

- Menu ကနေ option 5 ကို ရွေးပြီး product တစ်ခုကို ဖျက်ပါ။



6. Exit:

- Menu ကနေ option 6 ကို ရွေးပြီး program ကို ထွက်ပါ။

၂၂.၃။ Testing Results

--- Product CRUD Menu ---

1. Add Product
2. View All Products
3. View Product by ID
4. Update Product
5. Delete Product
6. Exit

Enter your choice (1-6): 1

Enter product name: Laptop

Enter product price: 1200.50

Enter product quantity: 10

Product added successfully!

Enter your choice (1-6): 2

ID: 1, Name: Laptop, Price: 1200.5, Quantity: 10

Enter your choice (1-6): 4

Enter product ID to update: 1

Enter new name (Laptop): Gaming Laptop

Enter new price (1200.5): 1500.75

Enter new quantity (10): 5

Product updated successfully!

Enter your choice (1-6): 2

ID: 1, Name: Gaming Laptop, Price: 1500.75, Quantity: 5

Enter your choice (1-6): 5

Enter product ID to delete: 1

Product deleted successfully!

Enter your choice (1-6): 2



No products found!

Enter your choice (1-6): 6

Exiting the program. Goodbye!

✍ Short Notes:

ဒီ program က Python နဲ့ SQLite3 ကို သုံးပြီး CRUD operations တွေကို လုပ်ဆောင်တဲ့ ဥပမာတစ်ခုဖြစ်ပါတယ်။ Project structure ကို သေချာစီစဉ်ထားတာကြောင့် code တွေကို လွယ်ကူစွာ maintain လုပ်နိုင်ပါတယ်။

စာရေးသူ ကိုယ်ရေးအကျဉ်း

ဤစာအုပ်ကိုရေးသားပြုစုသူကတော့ ကျွန်တော် ဆရာတင်မိုင်ဇော် ဖြစ်ပါတယ်။ ကျွန်တော်က မြစ်ကြီးနားမြို့ အထက (၁) မှာ အထက်တန်းအောင်မြင်ခဲ့ပါတယ်။ ပြီးတော့ ကျွန်တော် ဖိလိပိုင်နိုင်ငံ University of the Philippines (Los Banos) မှာ B.Sc Computer Science ကိုဆက်လက်ပညာသင်ယူခဲ့ပါတယ်။ ၂၀၀၄ မှာ အိုင်တီနဲ့ ဘွဲ့ရကျောင်းပြီးခဲ့ပါတယ်။ ၂၀၀၅ မှာ မြန်မာပြည်ပြန်လားပြီး မြစ်ကြီးနား ဇာတိမြို့မှာ Northern City Computer Training Center ကွန်ပျူတာသင်တန်းကျောင်းကို စတင်တည်ထောင်ဖွင့်လှစ်ခဲ့ပါတယ်။ လေ့လာဆည်းပူးခဲ့တဲ့ programming IT ဘာသာရပ်တွေကို သင်ကြားပို့ချခဲ့တာ ဖြစ်ပါတယ်။ ၂၀၀၉ မှာ မလေးရှားနိုင်ငံ Kuala Lumpur မှာ zepto it solution လို့ခေါ်တဲ့ အိုင်တီ Company မှာ Software Developer (programmer) အဖြစ် ၃ နှစ်တာ အလုပ်လုပ်ခဲ့ပါတယ်။ ကျန်းမာရေးအခြေနေကြောင့် ရန်ကုန်မြို့ကို ပြန်လာပြီး ဆေးကုသရင်း ရန်ကုန်မြို့မှာပဲ ValueStar Computer Institute မှာ ၆ နှစ်တာ IT Lecturer အနေနဲ့ရော IT Department Head အနေနဲ့ရော ဆက်လက်ခြေချခဲ့ပါတယ်။ ၂၀၁၇ မှာ ကိုယ်ပိုင် အိုင်တီသင်တန်းကျောင်းအဖြစ် Northern City Center နာမည်နဲ့ အရင်က မြစ်ကြီးနားမှာ တည်ထောင်ခဲ့ဖူး တဲ့ နာမည်ကို ပြန်လည်အသုံးပြုပြီး ဖွင့်လှစ်တည်ထောင်ခဲ့တာဖြစ်ပါတယ်။



References:

- "Automate the Boring Stuff with Python: Practical Programming for Total Beginners", Al Sweigart, 2nd Edition (2019)
- "Head-First Python: A Brain-Friendly Guide", Paul Barry, 1st Edition (2010)
- "Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code", Zed A. Shaw, 2017
- "Fluent Python: Clear, Concise, and Effective Programming", Luciano Ramalho, 2nd Edition (2022)
- "Effective Python: 90 Specific Ways to Write Better Python", Brett Slatkin, 2nd Edition (2019)
- "Python Cookbook: Recipes for Mastering Python 3", David Beazley & Brian K. Jones, 3rd Edition (2013)
- <https://www.w3schools.com/python>
- <https://www.sololearn.com/en/>
- <https://www.freecodecamp.org/>
- <https://deepseek.com>
- <https://openai.com/index/chatgpt/>

Appendix

စာရေးသူ၏ အိုင်တီအတွေ့အကြုံ မှတ်တမ်းများ -

Popular Web projects –

- Aya Internet Banking
<https://www.ayaibanking.com/>
- Japan Used Car Sales & Show room
<https://www.sbtjapan.com/sbt-myanmar/>
- Myanmar Traditional Boxing
<https://www.myanmartraditionalboxing.com.mm>
- Myanmar Agriculture Machinery & Products
<https://tptyeeshinn.com.mm>
- Real Estate
<https://www.saikhungnounge.com>
- Malaysia Minimart
<https://familystore.com.my>
- China Products & Fashion Sales
<https://youhome.space/>
- Online Payment System
<https://ucapay.com.mm>

Popular Point of Sales Application Projects –

- Malaysia Family Store Desktop Application and Mobile Application
- Myitkyina iGu Café & Gallery Desktop Application
- Myitkyina Hospital Blood Bank Desktop Application
- Myitkyina Fuji Food & Drinks Desktop Application
- Yangon Joyzone Stock Controller Desktop Application and Mobile Application
- Yangon Malihka Restaurant Desktop Application and Mobile Application
- Yangon Yuri Café Mobile Application and Mobile Application
- Yangon Gracevines Agarwood Desktop Application and Mobile Application

Documentary Photos



